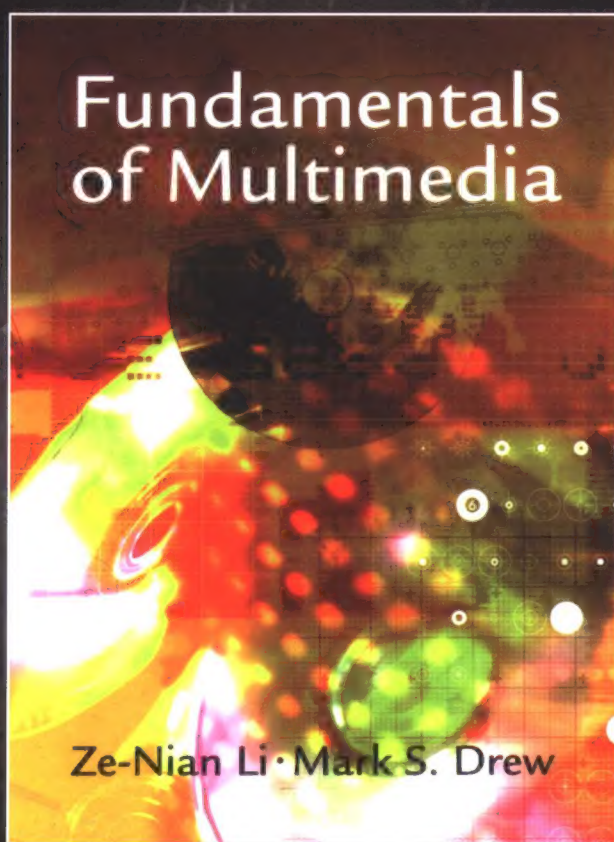


# 多媒体技术教程

(加) Ze-Nian Li Mark S. Drew 著 史元春 等译



Fundamentals of Multimedia

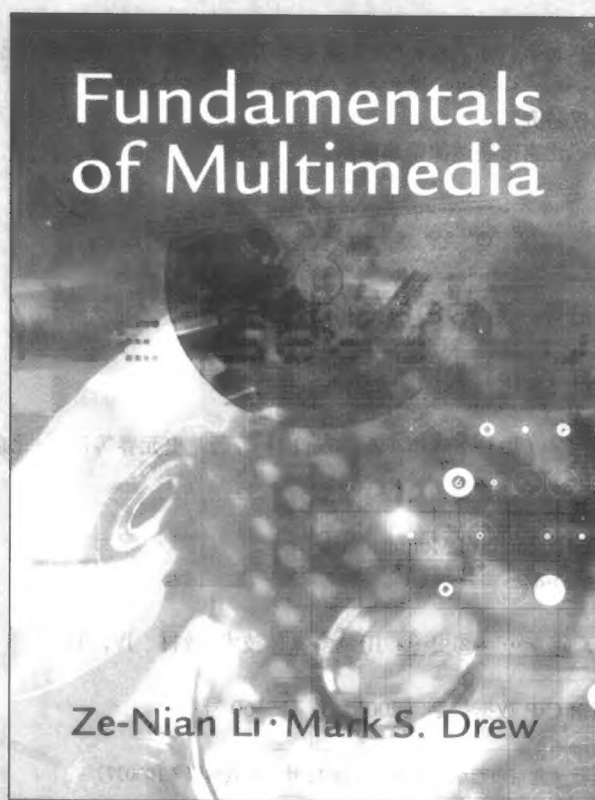


机械工业出版社  
China Machine Press

计 算 机 科 学 丛 书

# 多媒体技术教程

(加) Ze-Nian Li Mark S. Drew 著 史元春 等译



**Fundamentals of Multimedia**



机械工业出版社  
China Machine Press

本书从多媒体编著和数据表现、多媒体数据压缩以及多媒体通信和检索三个层面对多媒体涉及的基本概念、基本原理和基本技术进行了详细介绍。每章后包含和该章内容相关的网站和参考资源，并配有难易适当的课后练习。

本书内容全面，重点突出，适合作为高等院校多媒体技术课程的教材，也适合多媒体技术的研究人员和开发人员参考。

Simplified Chinese edition copyright © 2007 by Pearson Education Asia Limited and China Machine Press.

Original English language title: *Fundamentals of Multimedia* (ISBN: 0-13-061872-1) by Ze-Nian Li and Mark S. Drew, Copyright © 2004.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Prentice-Hall, Inc.

本书封面贴有 Pearson Education (培生教育出版集团) 激光防伪标签，无标签者不得销售。

版权所有，侵权必究。

本书法律顾问 北京市展达律师事务所

本书版权登记号：图字：01-2004-1213

### 图书在版编目 (CIP) 数据

多媒体技术教程 / (加) 李泽年 (Ze-Nian Li) 等著；史元春等译. -北京：机械工业出版社，2007.1

(计算机科学丛书)

书名原文：Fundamentals of Multimedia

ISBN 7-111-19975-8

I. 多… II. ①李… ②史… III. 多媒体技术-教材 IV. TP37

中国版本图书馆 CIP 数据核字 (2006) 第 115160 号

机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码 100037)

责任编辑：朱 劼 李南丰

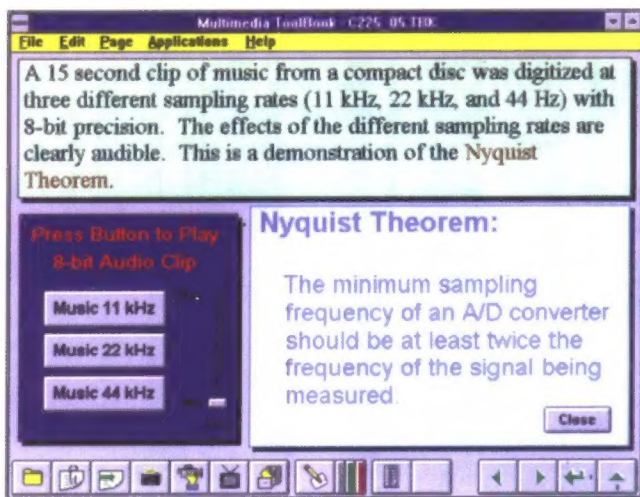
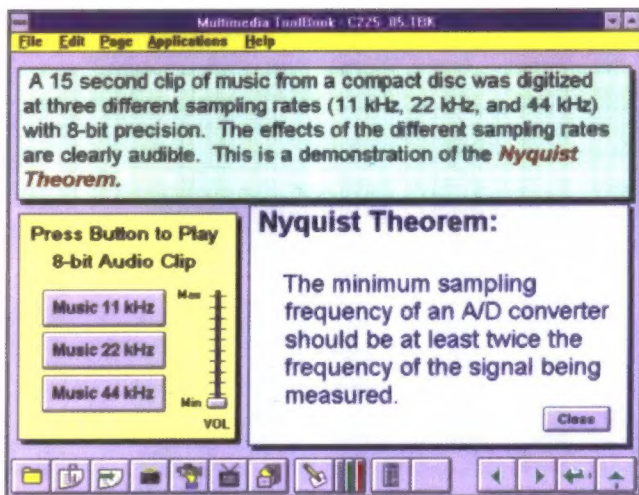
北京慧美印刷有限公司印刷·新华书店北京发行所发行

2007 年 1 月第 1 版第 1 次印刷

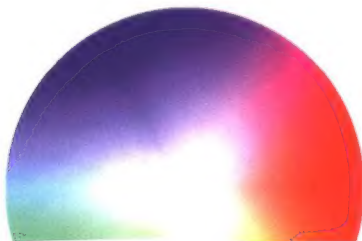
184mm×260mm·26.75 印张 (彩插 0.5 印张)

定价：42.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换  
本社购书热线：(010) 68326294



▲ 图 2-4 颜色和字体，感谢 Ron Vetter



▲ 图 2-6 调色板





a) 24 位彩色图像 forestfire.bmp



b) 图像的 R 颜色通道



c) 图像的 G 颜色通道



d) 图像的 B 颜色通道

▲ 图 3-5 高分辨率的彩色图像和单独的 R、G、B 颜色通道图像

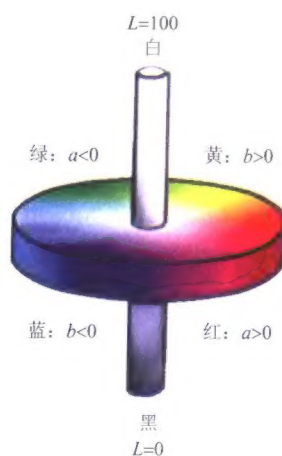


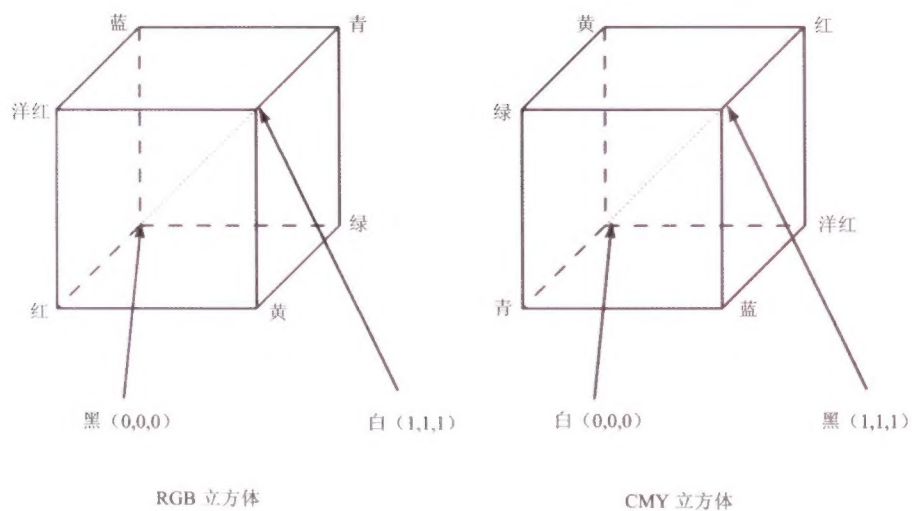
▲ 图 3-7 8 位彩色图像的例子



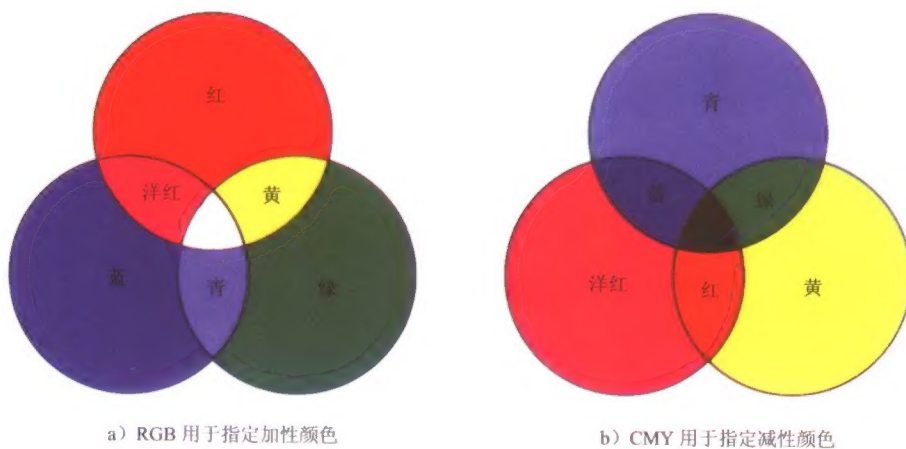
▲ 图 3-17 用户指定的低质量的 JPEG 图像

► 图 4-14 CIELAB 模型





▲ 图 4-15 RGB 和 CMY 颜色立方体



▲ 图 4-16 加性和减性颜色



a) 原始彩色图像



b)  $Y$

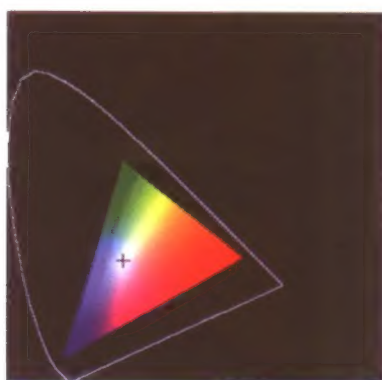


c)  $U$



d)  $V$

▲ 图 4-18 彩色图像的  $YUV$  分解



▲ 图 4-21 SMPTE 监视器色域



a) 原图



b) 以 0.75bpp 压缩的 JPEG 图像 (左) 和 JPEG 2000 图像 (右)



c) 以 0.25bpp 压缩的 JPEG 图像 (左) 和 JPEG 2000 图像 (右)

▲ 图 9-13 JPEG 和 JPEG 2000 的压缩





a) 背景的子图像全景图

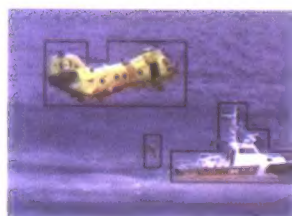


b) 蓝屏图中的前景对象（风笛手）

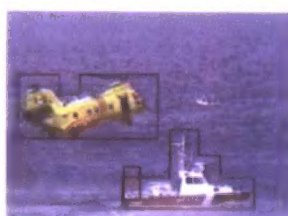


c) 合成的视频场景风笛手图像来自 Simon Fraser University 风笛乐队，在此感谢

▲ 图 12-10 子图像编码



运动区域：直升机

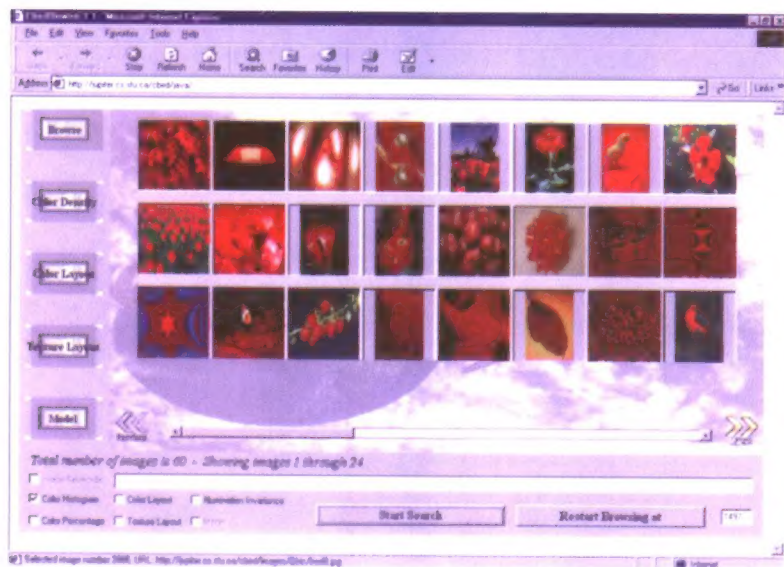


运动区域：人



运动区域：船

▲ 图 12-19 MPEG-7 视频段



▲ 图 18-3 颜色直方图的查找结果（一些小图片来自 Corel Gallery, Corel 拥有其版权）



▲ 图 18-8 C-BIRD 界面，显示使用基本的椭圆进行对象选择（本图片来自 Corel Gallery, Corel 拥有其版权）



a) 示例模型图像



b) 示例数据库图像，其中包含模型书

◀ 图 18-10 模型和目标图像  
(Active Perception 的封面来自  
Lawrence Erlbaum Associates  
公司，在此感谢!)

► 图 18-13 颜色场景



a) 模型图像的颜色场景



b) 数据库图像的颜色场景

# 出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域中取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅筹划了研究的范畴，还揭开了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall, Addison-Wesley, McGraw-Hill, Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum, Stroustrup, Kernighan, Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及度藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总体规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师提供服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业



的教学度身订造的。其中许多教材均已为M. I. T., Stanford, U.C. Berkeley, C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程,而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下,读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑,这些因素使我们的图书有了质量的保证,但我们的目标是尽善尽美,而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正,我们的联系方式如下:

电子邮件: [hzjsj@hzbook.com](mailto:hzjsj@hzbook.com)

联系电话: (010) 68995264

联系地址: 北京市西城区百万庄南街1号

邮政编码: 100037

# 专家指导委员会

(按姓氏笔画顺序)

尤晋元  
石教英  
张立昂  
邵维忠  
周克定  
郑国梁  
高传善  
裘宗燕

王 珊  
吕 建  
李伟琴  
陆丽娜  
周傲英  
施伯乐  
梅 宏  
戴 葵

冯博琴  
孙玉芳  
李师贤  
陆鑫达  
孟小峰  
钟玉琢  
程 旭

史忠植  
吴世忠  
李建中  
陈向群  
岳丽华  
唐世渭  
程时端

史美林  
吴时霖  
杨冬青  
周伯生  
范 明  
袁崇义  
谢希仁

## 译者序

如今,多媒体这个词已经不是十多年前那个笼罩着神秘色彩的专业术语了,音视频等基本多媒体功能部件已经成为计算机必要且廉价的组成部分。多媒体技术也逐渐成为计算机学科的必修课程之一,从专科课程、本科课程到研究生课程都有多种课程设置,这是一个有趣的现象,为什么呢?

这个现象不但生动体现了技术与应用相互促进的关系,而且深刻反映了多媒体对计算机技术与应用的强烈冲击。计算机的处理对象经历了从数值、文字到多媒体的发展历程,计算机也相应地扮演着实验室计算工具、办公室文字处理工具和个人信息交流工具的不同角色,多媒体给人们带来生动体验的同时,也对计算机的处理能力提出了全方位的要求。以图形图像、音视频为代表的多媒体,要求海量的存储、高速的计算、即时的通信、便捷的交互,对计算机系统的信息表示、计算处理、存储检索、网络通信、人机交互提出了全面和震动性的技术挑战,成为从 20 世纪 80 年代末以来的一个持续的热点研究领域。

然而,正是由于多媒体技术涉及的内容广泛,使得多媒体技术课程的内容既可散布于计算机专业的一些传统课程中,也可以因授课对象、选题重点、应用与研究等不同的考虑形成新的不同课程,因此,多媒体技术相关的书很多,教材的选择倒成了难题。

我认为,加拿大 Simon Fraser 大学的 Ze-Nian Li (李泽年) 和 Mark S. Drew 两位编写的这本书是一本很难得的好教材,因为它具有以下特点:

1) 内容全面,涉及多媒体数据表示与编著、数据编码、压缩、通信、检索等,教师可以有选择地采用其中的内容;

2) 原理突出,能在有限的篇幅内从繁杂的内容中将基本原理阐述清楚,这源于作者在该领域多年的研究工作和对该领域的深刻理解;

3) 练习得当,做每章后面的习题是学习掌握相关内容的必要环节,本书的练习难易适当、操作性强;

4) 索引丰富,不但有分门别类的网站和参考书的索引,而且每章后的“进一步探索”很能启发学生的思维。成就这本优秀教材的是两位作者近 10 年的多媒体技术课程的教学经验,可以说这些教材内容真正来源于课堂,所以对内容的剪裁适当。

当然,对多媒体技术的研究者与开发人员来讲,本书也是一本非常有用的技术参考书,它全面、适当、翔实,适合作为课外学习与研究应用的参考。

由于工作繁忙,翻译时间拖得比较长,非常感谢出版社对我的体谅。由于时间和能力的限制,若有错误和不妥之处敬请读者指正。

参加本书翻译的还有:贺伟晟、肖鑫、毕小俊、常斌、李晶瑜、郭玲、张南、陈全斌、唐硕、姜皓、龚伟。谢谢他们。

史元春

2005 年于清华大学计算机系

# 前 言

多媒体课程逐渐成为计算机科学和计算机工程学科的必修课程之一，尤其是现在，多媒体涉及计算机科学的众多领域。多媒体最初被认为是一个纵向应用领域，也就是说，它有一套独有的方法。然而，像普适计算一样，多媒体现在已经成为一个横向的应用领域，并且是许多学科的重要研究内容，如计算机图形学、图像处理、数据库、实时系统、操作系统、信息检索、计算机网络、计算机视觉等。多媒体不再是一种玩具，而是成为我们进行工作、思考的技术环境的重要组成部分。本书面向大学多媒体教学的需求，介绍计算机科学发展历程中与多媒体相关的部分，以及与多媒体相关的计算机科学和工程领域的其他问题。

本书不是介绍简单的设计问题，而是面向更高级的读者；也不是一本参考书，更像是一本传统意义上的教材。虽然我们必然要在书中讨论多媒体工具，但更多地是讲述这些工具的工作原理。学完本书的读者能够真正掌握多媒体最基础的原理。

本书内容丰富，能够使學生利用这些知识开展多媒体上的有趣而奇妙的实践项目和交互式项目，甚至能有志于传授这些概念。

## 本书读者对象

本书介绍多媒体领域的基本知识，定位于计算机科学与计算机工程专业的学生。本书的目标读者是本科高年级的学生，也能用于更高年级的课程；对于想了解最新多媒体技术的专业人员，以及任何有兴趣的人，它也是一本很好的参考书。对于没有接触过本学科的研究生，如果想拥有扎实的基础，阅读本书会必定会受益。

本书重点介绍概念，而不是应用。在多媒体课上，教师将教授概念、测试概念，但同样也让学生用已有的编程技巧来解决多媒体问题。本书的配套网站给出了一些多媒体应用的代码，以及精选的课程设计和其他有用的资源。

本书介绍的概念会在课程设计中得到体现。我们假设读者会编程，而且乐于学习使用新的编程工具。本书不把重点放在工具讲解上，而是强调学生不要只会用工具。应用本书所介绍的方法和思想，学生能够通过自学学到更多，一种方式是通过设定的工作场景。利用本书学习多媒体课程的学生，能够在四年级或者更早的时候开始从事多媒体相关工作，这并不稀奇。

本书选择的内容包括读者在实际应用中会遇到的问题。有些内容比较简单但比较新；有些内容比较复杂，但对这个新兴的领域来说，这是不可避免的。

## 作者在实际教学中是否使用过本书

从 1996 年开始，我们开始教授本科生三年级的多媒体系统课程，我们使用的介绍性材料正是本书的前身。在一个学期的时间里可能不能讲授完本书的所有内容，我们通常安排讲授第一、二部分中大部分内容，加上简单介绍第三部分的某些章节。

至于更高级的介绍性课程，我们用本书的资料上过为期一学期的研究生课程。本科生四年级或者研究生的课程可以考虑讲授本书的第一、二部分内容，以及第三部分的某些内容，同时可以使用本书包括的研究文献和相关会议的研究结果。

我们试图满足本科生和研究生的要求，主要针对本科生，但涵盖一些更高级的内容。标有星



号的章节在初次阅读时可以跳过。

## 本书包含的内容

第一部分介绍多媒体技术相关的概念、发展历史和现状。特别要指出的是，因为要使用软件工具完成多媒体作业，所以这部分对多媒体工具进行概述，并且讲述多媒体著作工具的细节。数据表现对多媒体很重要，本书将介绍多媒体应用中最重要数据表现方式，详细讨论图像数据、视频数据、音频数据。颜色对多媒体程序是至关重要的，这部分将介绍颜色对多媒体的影响和作用。

第二部分介绍如何在屏幕上和扬声器中播多媒体数据。数据压缩是多媒体广泛应用的重要技术，这部分将介绍无损压缩和有损压缩技术。对后者，JPEG 静止图像压缩标准（包括 JPEG 2000）显然是最重要的，所以会详细介绍 JPEG。但因为静止图形很重要，而视频更重要，因此这部分还介绍 MPEG 系列标准：MPEG-1、MPEG-2、MPEG-4、MPEG-7 以及其他更高版本。另外，第二部分还介绍了基本的音频压缩技术，简要讲述 MPEG 音频，其中包括 MP3。

第三部分介绍多媒体技术对网络和系统的种种需求。接着介绍使交互式多媒体成为可能的网络技术和协议。“媒体点播”、“IP 上的多媒体”、“ATM 上的多媒体”、“无线网络上的多媒体”都在第三部分加以介绍。基于内容的检索对于数字图书馆和交互式多媒体尤其重要，因此这部分还将详细介绍与此相关的思想和系统。

## 本书的网站

本书的网站是 [www.cs.sfu.ca/mmbook](http://www.cs.sfu.ca/mmbook)。网站中包括本书的图片、勘误表、教学用的程序以及动态更新的每章的参考资料的链接。由于这些链接本身是不断更新的（当然 URL 也会变动），因此以在线方式而不是纸质文本提供。

## 教师资源

Prentice Hall 有一个网站提供相关教学资源，包括众多讲义、一学期课时教学大纲、教学安排、练习答案、示例作业和解答、示例测试卷以及更多的测试试题。需要这些资源的教师可填写书后的“教学支持说明”并与出版社联系。

## 致谢

我们对审阅本书的同事表示衷心的感谢。他们是 Shu-Ching Chen、Edward Chang、Qianping Gu、Rachelle S. Heller、Gongzhu Hu、S. N. Jayaram、Tiko Kameda、Xiaobo Li、Siwei Lu、Dennis Richards 以及 Jacques Vaisey。

本书在编写过程中，我们现在及过去的很多同事和学生给出了很好的建议。我们对 James Au、Chad Ciavarro、Hao Jiang、Steven Kithau、Michael King、Cheng Lu、Yi Sun、Dominic Szopa、Zinovi Tauber、Malte von Ruden、Jian Wang、Jie Wei、Edward Yan、Yingchen Yang、Osmar Zaiane、Wenbiao Zhang、William Zhong 表示感谢。Ye Lu 先生对本书第 8、9 章做出了重要贡献，我们对他表示特别感谢。对于为完善本书而调试成功课程设计（Student Projects）的学生们，我们同样致以深深的谢意。

# 目 录

出版者的话  
专家指导委员会  
译者序  
前言

## 第一部分 多媒体编著和数据表现

第 1 章 多媒体导论 .....	2
1.1 什么是多媒体 .....	2
1.1.1 多媒体的组成部分 .....	2
1.1.2 多媒体的研究课题和研究项目 .....	3
1.2 多媒体和超媒体 .....	3
1.2.1 多媒体的历史 .....	4
1.2.2 超媒体和多媒体 .....	5
1.3 World Wide Web .....	5
1.3.1 WWW 的历史 .....	5
1.3.2 HTTP .....	6
1.3.3 HTML .....	7
1.3.4 XML .....	7
1.3.5 SMIL .....	8
1.4 多媒体软件工具概述 .....	9
1.4.1 编曲和谱曲 .....	10
1.4.2 数字音频 .....	10
1.4.3 图形和图像编辑 .....	10
1.4.4 视频编辑 .....	11
1.4.5 动画 .....	11
1.4.6 多媒体编著 .....	12
1.5 进一步探索 .....	12
1.6 练习 .....	13
1.7 参考文献 .....	13
第 2 章 多媒体编著和工具 .....	15
2.1 多媒体编著 .....	15
2.1.1 多媒体编著的模式 .....	15
2.1.2 多媒体作品 .....	17
2.1.3 多媒体展现 .....	18
2.1.4 自动编著 .....	22

2.2 多媒体编辑和编著工具 .....	25
2.2.1 Adobe Premiere .....	25
2.2.2 Macromedia Director .....	28
2.2.3 Macromedia Flash .....	31
2.2.4 Dreamweaver .....	34
2.3 VRML .....	34
2.3.1 概述 .....	34
2.3.2 动画和交互 .....	36
2.3.3 VRML 规范 .....	37
2.4 进一步探索 .....	37
2.5 练习 .....	38
2.6 参考文献 .....	39
第 3 章 图形和图像的数据表现 .....	41
3.1 图形/图像的数据类型 .....	41
3.1.1 1 位图像 .....	41
3.1.2 8 位灰度图像 .....	41
3.1.3 图像数据类型 .....	44
3.1.4 24 位彩色图像 .....	44
3.1.5 8 位彩色图像 .....	44
3.1.6 颜色查找表 .....	46
3.2 常见的文件格式 .....	48
3.2.1 GIF .....	48
3.2.2 JPEG .....	51
3.2.3 PNG .....	52
3.2.4 TIFF .....	52
3.2.5 EXIF .....	52
3.2.6 图形动画文件 .....	52
3.2.7 PS 和 PDF .....	52
3.2.8 Windows WMF .....	53
3.2.9 Windows BMP .....	53
3.2.10 Macintosh PAINT 和 PICT .....	53
3.2.11 X Windows PPM .....	53
3.3 进一步探索 .....	53
3.4 练习 .....	54
3.5 参考文献 .....	55

第 4 章 图像和视频中的颜色 .....	56	5.2.3 SECAM 视频 .....	82
4.1 颜色科学 .....	56	5.3 数字视频 .....	82
4.1.1 光和光谱 .....	56	5.3.1 色度的二次采样 .....	82
4.1.2 人的视觉 .....	57	5.3.2 数字视频的 CCIR 标准 .....	83
4.1.3 眼睛的光谱灵敏度 .....	57	5.3.3 HDTV .....	84
4.1.4 图像的形成 .....	58	5.4 进一步探索 .....	85
4.1.5 照相机系统 .....	59	5.5 练习 .....	85
4.1.6 伽马校正 .....	59	5.6 参考文献 .....	86
4.1.7 颜色匹配函数 .....	61	第 6 章 数字音频基础 .....	87
4.1.8 CIE 色度图 .....	62	6.1 声音数字化 .....	87
4.1.9 彩色显示器规范 .....	64	6.1.1 什么是声音 .....	87
4.1.10 超色域的颜色 .....	65	6.1.2 数字化 .....	87
4.1.11 白点校正 .....	65	6.1.3 奈奎斯特理论 .....	88
4.1.12 XYZ 到 RGB 的转换 .....	66	6.1.4 信噪比 .....	90
4.1.13 带伽马校正的转换 .....	66	6.1.5 信号量化噪声比 .....	90
4.1.14 $L^*a^*b^*$ (CIELAB)颜色模型 .....	67	6.1.6 线性量化和非线性量化 .....	91
4.1.15 其他颜色坐标系统 .....	68	6.1.7 音频滤波 .....	93
4.1.16 蒙塞尔颜色命名系统 .....	68	6.1.8 音频质量与数据率 .....	93
4.2 图像中的颜色模型 .....	68	6.1.9 合成的声音 .....	94
4.2.1 CRT 显示器的颜色模型 .....	68	6.2 MIDI: 乐器数字化接口 .....	95
4.2.2 减色法: CMY 颜色模型 .....	69	6.2.1 MIDI 概述 .....	95
4.2.3 从 RGB 到 CMY 的转换 .....	69	6.2.2 MIDI 硬件 .....	97
4.2.4 消除不足颜色: CMYK 系统 .....	69	6.2.3 MIDI 消息的结构 .....	98
4.2.5 打印机色域 .....	70	6.2.4 通用 MIDI .....	101
4.3 视频中的颜色模型 .....	71	6.2.5 MIDI 到 WAV 的转换 .....	101
4.3.1 视频颜色转换 .....	71	6.3 音频的量化和传输 .....	101
4.3.2 YUV 颜色模型 .....	71	6.3.1 音频的编码 .....	101
4.3.3 YIQ 颜色模型 .....	72	6.3.2 脉冲编码调制 .....	101
4.3.4 YCbCr 颜色模型 .....	73	6.3.3 音频的差分编码 .....	103
4.4 进一步探索 .....	73	6.3.4 无损预测编码 .....	104
4.5 练习 .....	74	6.3.5 DPCM .....	106
4.6 参考文献 .....	76	6.3.6 DM .....	108
第 5 章 视频中的基本概念 .....	77	6.3.7 ADPCM .....	109
5.1 视频信号的类型 .....	77	6.4 进一步探索 .....	110
5.1.1 分量视频 .....	77	6.5 练习 .....	111
5.1.2 复合视频 .....	77	6.6 参考文献 .....	112
5.1.3 S-Video .....	78		
5.2 模拟视频 .....	78		
5.2.1 NTSC 视频 .....	79		
5.2.2 PAL 视频 .....	81		
		第二部分 多媒体数据压缩	
		第 7 章 无损压缩算法 .....	117
		7.1 简介 .....	117

7.2 信息论基础 .....	118	9.1.1 JPEG 图像压缩的主要步骤 .....	176
7.3 游长编码 .....	120	9.1.2 JPEG 模式 .....	182
7.4 变长编码 .....	120	9.1.3 JPEG 位流概述 .....	184
7.4.1 香农-凡诺算法 .....	120	9.2 JPEG 2000 标准 .....	184
7.4.2 赫夫曼编码 .....	122	*9.2.1 JPEG 2000 图像压缩的主要 步骤 .....	185
7.4.3 自适应赫夫曼编码 .....	124	9.2.2 使 EBCOT 适合 JPEG 2000 .....	190
7.5 基于字典的编码 .....	127	9.2.3 感兴趣区域编码 .....	191
7.6 算术编码 .....	131	9.2.4 JPEG 和 JPEG 2000 的性能 比较 .....	192
7.7 无损图像压缩 .....	134	9.3 JPEG-LS 标准 .....	193
7.7.1 图像的差分编码 .....	134	9.3.1 预测 .....	194
7.7.2 无损 JPEG .....	135	9.3.2 确定上下文 .....	194
7.8 进一步探索 .....	136	9.3.3 残差编码 .....	195
7.9 练习 .....	136	9.3.4 准无损模式 .....	195
7.10 参考文献 .....	137	9.4 二值图像压缩标准 .....	195
第 8 章 有损压缩算法 .....	139	9.4.1 JBIG 标准 .....	195
8.1 简介 .....	139	9.4.2 JBIG2 标准 .....	196
8.2 失真量度 .....	139	9.5 进一步探索 .....	197
8.3 比率失真理论 .....	140	9.6 练习 .....	197
8.4 量化 .....	140	9.7 参考文献 .....	199
8.4.1 均匀标量量化 .....	140	第 10 章 基本视频压缩技术 .....	200
8.4.2 非均匀标量量化 .....	142	10.1 视频压缩简介 .....	200
*8.4.3 矢量量化 .....	143	10.2 基于运动补偿的视频压缩 .....	200
8.5 变换编码 .....	144	10.3 搜索运动向量 .....	201
8.5.1 离散余弦变换 .....	144	10.3.1 顺序搜索 .....	201
*8.5.2 Karhunen-Loève 变换 .....	154	10.3.2 2D 对数搜索 .....	202
8.6 小波编码 .....	156	10.3.3 分层搜索 .....	203
8.6.1 简介 .....	156	10.4 H.261 .....	205
*8.6.2 连续小波变换 .....	159	10.4.1 I 帧编码 .....	206
*8.6.3 离散小波变换 .....	161	10.4.2 P 帧预测编码 .....	206
8.7 小波包 .....	167	10.4.3 H.261 的量化 .....	207
8.8 小波系数的嵌入零树 .....	168	10.4.4 H.261 编码器和解码器 .....	207
8.8.1 零树数据结构 .....	168	10.4.5 H.261 视频位流语法概述 .....	209
8.8.2 逐次近似量化 .....	169	10.5 H.263 .....	210
8.8.3 EZW 示例 .....	170	10.5.1 H.263 的运动补偿 .....	211
8.9 层次树的集合划分 .....	172	10.5.2 H.263 的可选编码模式 .....	212
8.10 进一步探索 .....	172	10.5.3 H.263+和 H.263++ .....	213
8.11 练习 .....	172	10.6 进一步探索 .....	214
8.12 参考文献 .....	174	10.7 练习 .....	214
第 9 章 图像压缩标准 .....	176		
9.1 JPEG 标准 .....	176		



10.8 参考文献 .....	215	12.7 MPEG-21 .....	256
第 11 章 MPEG 视频编码 I: MPEG-1 和 MPEG-2 .....	216	12.8 进一步探索 .....	256
11.1 概述 .....	216	12.9 练习 .....	256
11.2 MPEG-1 .....	216	12.10 参考文献 .....	257
11.2.1 MPEG-1 中的运动补偿 .....	216	第 13 章 音频压缩技术基础 .....	260
11.2.2 与 H.261 的其他主要区别 .....	218	13.1 语音编码中的 ADPCM .....	260
11.2.3 MPEG-1 视频位流 .....	220	ADPCM .....	260
11.3 MPEG-2 .....	221	13.2 G.726 ADPCM .....	261
11.3.1 支持隔行扫描视频 .....	221	13.3 声音合成器 .....	262
11.3.2 MPEG-2 的可伸缩性 .....	224	13.3.1 相位不敏感性 .....	262
11.3.3 与 MPEG-1 的其他主要区别 .....	228	13.3.2 通道声音合成器 .....	263
11.4 进一步探索 .....	228	13.3.3 共振峰声音合成器 .....	264
11.5 练习 .....	228	13.3.4 线性预测编码 .....	264
11.6 参考文献 .....	229	13.3.5 CELP .....	266
第 12 章 MPEG 视频编码 II: MPEG-4、 MPEG-7 及更高版本 .....	230	*13.3.6 混合激励声音合成器 .....	270
12.1 MPEG-4 概述 .....	230	13.4 进一步探索 .....	272
12.2 MPEG-4 的基于对象的视觉编码 .....	232	13.5 练习 .....	273
12.2.1 基于 VOP 的编码与基于帧的 编码 .....	232	13.6 参考文献 .....	274
12.2.2 运动补偿 .....	233	第 14 章 MPEG 音频压缩 .....	275
12.2.3 纹理编码 .....	236	14.1 心理声学简介 .....	275
12.2.4 形状编码 .....	238	14.1.1 等响度关系 .....	275
12.2.5 静态纹理编码 .....	239	14.1.2 频率遮掩 .....	277
12.2.6 子图像编码 .....	240	14.1.3 时间遮掩 .....	280
12.2.7 全局运动补偿 .....	240	14.2 MPEG 音频 .....	282
12.3 MPEG-4 的合成对象编码 .....	241	14.2.1 MPEG 的层 .....	282
12.3.1 2D 网格对象编码 .....	241	14.2.2 MPEG 音频策略 .....	282
12.3.2 基于模型的 3D 编码 .....	246	14.2.3 MPEG 音频压缩算法 .....	283
12.4 MPEG-4 对象类型、规格和等级 .....	247	14.2.4 MPEG-2 高级音频编码 .....	287
12.5 MPEG-4 Part10/H.264 .....	248	14.2.5 MPEG-4 音频 .....	288
12.5.1 核心特征 .....	249	14.3 其他商业音频编解码器 .....	289
12.5.2 基本规格特征 .....	250	14.4 未来: MPEG-7 和 MPEG-21 .....	289
12.5.3 主要规格特征 .....	250	14.5 进一步探索 .....	289
12.5.4 扩展规格特征 .....	250	14.6 练习 .....	290
12.6 MPEG-7 .....	250	14.7 参考文献 .....	291
12.6.1 描述子 .....	252		
12.6.2 描述方案 .....	253	第三部分 多媒体通信和检索	
12.6.3 描述定义语言 .....	255	第 15 章 计算机和多媒体网络 .....	294
		15.1 计算机和多媒体网络基础 .....	294
		15.1.1 OSI 网络的层次 .....	294
		15.1.2 TCP/IP 协议 .....	294

15.2 多路复用技术.....	297	16.7 练习.....	333
15.2.1 多路复用技术基础.....	297	16.8 参考文献.....	334
15.2.2 综合业务数字网络.....	298	第 17 章 无线网络.....	336
15.2.3 同步光纤网络.....	299	17.1 简介.....	336
15.2.4 非对称数字用户线路.....	300	17.1.1 模拟无线网络.....	336
15.3 LAN 和 WAN.....	301	17.1.2 数字无线网络.....	337
15.3.1 局域网.....	301	17.1.3 TDMA 和 GSM.....	337
15.3.2 广域网.....	303	17.1.4 扩频和 CDMA.....	339
15.3.3 异步传输模式.....	304	17.1.5 CDMA 分析.....	341
15.3.4 千兆和 10 千兆以太网.....	306	17.1.6 3G 数字无线网络.....	343
15.4 接入网.....	306	17.1.7 无线局域网.....	345
15.5 通用外设接口.....	307	17.2 无线电传播模式.....	347
15.6 进一步探索.....	308	17.2.1 多径衰减.....	347
15.7 练习.....	308	17.2.2 路径损耗.....	348
15.8 参考文献.....	308	17.3 无线网络上的多媒体.....	349
第 16 章 多媒体网络通信和应用.....	309	17.3.1 同步损失.....	349
16.1 多媒体数据传输的质量.....	309	17.3.2 错误修复熵编码.....	350
16.1.1 服务质量.....	309	17.3.3 错误隐藏.....	352
16.1.2 IP 协议的 QoS.....	311	17.3.4 前向纠错.....	353
16.1.3 具有优先级的发送.....	311	17.3.5 无线交互式多媒体的趋势.....	355
16.2 IP 上的多媒体.....	312	17.4 进一步探索.....	357
16.2.1 IP 多播.....	312	17.5 练习.....	357
16.2.2 实时传输协议 (RTP).....	314	17.6 参考文献.....	358
16.2.3 实时控制协议 (RTCP).....	315	第 18 章 数字图书馆中基于内容的	
16.2.4 资源预留协议 (RSVP).....	315	检索.....	359
16.2.5 实时流协议 (RTSP).....	316	18.1 如何检索图像.....	359
16.2.6 因特网电话技术.....	317	18.2 C-BIRD: 一个实例研究.....	360
16.3 ATM 网上的多媒体.....	320	18.2.1 C-BIRD 的 GUI.....	361
16.3.1 ATM 网上的视频码率.....	320	18.2.2 颜色直方图.....	361
16.3.2 ATM 适配层.....	321	18.2.3 颜色密度.....	363
16.3.3 MPEG-2 会聚到 ATM.....	322	18.2.4 颜色分布.....	363
16.3.4 ATM 上的多播.....	322	18.2.5 纹理分布.....	364
16.4 MPEG-4 的传输.....	323	18.2.6 按光源恒常性查找.....	365
16.4.1 MPEG-4 中的 DMIF.....	323	18.2.7 按对象模型查找.....	366
16.4.2 IP 上的 MPEG-4.....	323	18.3 当前图像查找系统概览.....	377
16.5 媒体点播.....	324	18.3.1 QBIC.....	377
16.5.1 交互式电视和机顶盒.....	324	18.3.2 加州大学圣芭芭拉分校的	
16.5.2 视频点播的广播方案.....	325	搜索引擎.....	378
16.5.3 缓冲区管理.....	330	18.3.3 加州大学伯克利分校的数字	
16.6 进一步探索.....	333	图书馆项目.....	378

18.3.4 Chabot .....378

18.3.5 Blobworld .....378

18.3.6 哥伦比亚大学的图像  
搜索器 .....378

18.3.7 Informedia.....379

18.3.8 MetaSEEk .....379

18.3.9 Photobook 和 FourEyes .....379

18.3.10 MARS.....379

18.3.11 Virage .....380

18.3.12 VIPER.....380

18.3.13 Visual RetrievalWare .....380

18.4 相关反馈.....380

18.4.1 MARS .....380

18.4.2 iFind.....382

18.5 结果的量化 .....382

18.6 视频查询.....382

18.7 其他格式的查询.....385

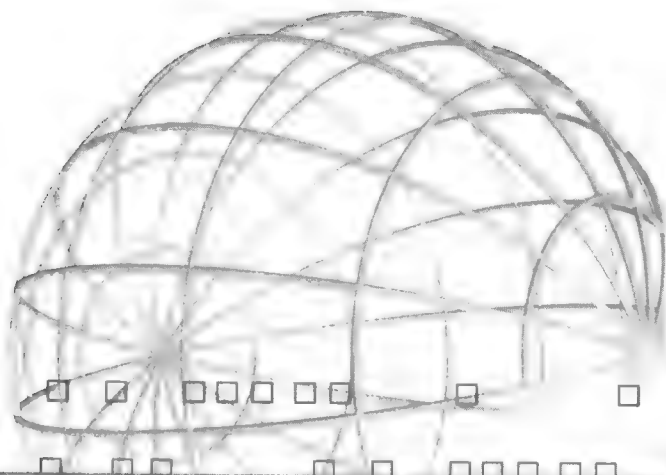
18.8 基于内容检索的展望 .....385

18.9 进一步探索 .....385

18.10 练习 .....386

18.11 参考文献 .....387

索引 .....391



# 第一部分 多媒体编著和数据表现

- 第1章 多媒体导论
- 第2章 多媒体编著和工具
- 第3章 图形和图像的数据表现
- 第4章 图像和视频中的颜色
- 第5章 视频中的基本概念
- 第6章 数字音频基础

## 多媒体导论

作为多媒体的导论，第1章围绕“什么是多媒体”进行论述。我们将介绍多媒体的发展历史和超文本与超媒体的演变过程，然后详细讨论与实践问题有关的多媒体软件工具，这些工具是我们创建多媒体内容的基本手段。然而，多媒体作品还远非这些部分的简单相加，因此，第2章讨论多媒体编著设计的基本要素并对编著方法加以分类。该章还列出当前一些重要的多媒体编著工具。

## 多媒体数据表现

和许多领域一样，如何最好地表现数据也是多媒体研究中的一个至关重要的问题。第3章~第6章讨论的就是这一问题，其中列举了多媒体应用中最重要的一些数据表现。因为最受关注的是图像、运动图片和音频，所以我们在第3章中开始讨论图形和图像的数据表现，然后在第5章继续讨论视频中的基本概念。在开始第6章的讨论之前，我们先在第4章讨论颜色使用的一些问题，因为颜色在多媒体程序中尤为重要。

# 第1章 多媒体导论

## 1.1 什么是多媒体

人们在使用“多媒体”这个术语时，往往对这个术语有不尽相同甚至截然相反的理解。PC 销售商希望我们将多媒体理解成这样的一台 PC：具有音效功能，带有 DVD 光驱，也许还有能够理解附加的多媒体指令的性能强劲的微处理器。而娱乐消费产品的销售商则会将多媒体理解为具有上百个频道的交互式有线电视，或者是通过高速 Internet 连接提供的类似于有线电视的服务。

计算机学科的学生则会从面向应用的角度理解多媒体：多媒体是由使用多模态技术（包括文本、图像、图形、动画、视频和音频等，以及交互活动）的应用程序构成的。在文化领域中非常流行的“融合”观点，在科学界同样广为接受。这一观点反映在多媒体领域，则是 PC、DVD、游戏设备、数字电视、通过机顶盒进行网络浏览以及无线传输等多种技术的融合，也许在不久的将来，这种融合就可以进一步扩展为功能全面的多媒体产品。硬件技术的提高将不断推动这类设备的发展，而现有的成果已经令人心动——多媒体业已成为计算机科学领域中最令人感兴趣的一个部分。很多过去单独研究的内容，在多媒体这个新的领域找到了共同点，进而促进了这种融合。图形、可视化、HCI、计算机视觉、数据压缩、图论、网络和数据库系统——都将对目前的多媒体的发展产生重要的影响。

### 1.1.1 多媒体的组成部分

多媒体中的文本、音频、图像、图形、动画和视频等多模态技术在以下领域中得到广泛应用：

- 视频会议。
- 高等教育中的远程教学。
- 远程医疗。
- 协同工作环境，在该环境下，可以支持用户对共享文档进行编辑或是共同参与一个电脑游戏。
- 在大信息量视频和图像数据库中对目标可视对象进行搜索。
- “增强”现实：在场景中加入具有真实形象的计算机图形和视频对象，以便考虑对象和光线的物理特性。
- 用音频信息对视频会议的参与者进行定位，并考虑参与者的注视方向和注意力。
- 在新视频中构建可搜索的特征，支持多种不同码率的全新的、可伸缩的多媒体产品。
- 可编辑（editable）的多媒体组件，即允许用户自行决定哪些组件、视频或图形是可见的，并允许用户对组件进行移动或删除等操作，并使组件具有分布式的结构。
- 建立“逆好莱坞”式的应用程序，用以重现视频产生的过程，并使用情节串连图板来删节和简化视频的内容。
- 使用语音识别技术建立交互环境——例如嵌入在厨房墙壁中的 Web 浏览器。

从计算机专业人员的角度来看，多媒体技术之所以有如此大的吸引力，是因为很多传统计算机科学领域中的研究内容都与它具有某种关联，如网络、操作系统、实时系统、视觉和信息检索等等。与数据库技术类似，多媒体和很多传统领域都产生了交叉。

### 1.1.2 多媒体的研究课题和研究项目

对于计算机科学领域的研究者来说,多媒体包括广泛的研究课题[1]:

- **多媒体处理和编码** 其中包括多媒体内容的分析,基于内容的多媒体检索,多媒体安全,音频/图像/视频的处理和压缩等。
- **多媒体系统支持和网络** 人们将这类问题理解为网络协议、Internet、操作系统、客户机和服务器、服务质量(QoS)和数据库。
- **多媒体工具、端系统和应用程序** 其中包括超媒体系统、用户接口、编著系统、多模态交互和集成:“无所不在性”——可以随时随地上网的设备、多媒体教育,包括计算机支持的学习和设计以及虚拟环境中的应用程序。

多媒体领域的研究同样影响着计算机科学的其他分支。例如,数据挖掘是目前一个重要的研究领域,而包含多媒体数据对象的大型数据库正是该领域研究的课题。远程医疗应用程序(例如“远程病人诊断咨询”系统)是对现有的网络构架提出严峻考验的多媒体应用程序。

4

#### 目前的多媒体研究项目

目前已经有不少多媒体的研究项目,下面将简单介绍其中的一些项目。

基于摄像头的对象跟踪技术始终是一个重要的研究内容。诸如工业控制或游戏这样的系统对于真实环境(例如棋盘游戏)中的运动模型(玩具)往往有着很强的依赖性,对象跟踪技术的一个目标就是为此类应用开发控制系统。对控制对象(玩具)进行跟踪就可以使用户对整个过程进行控制。

3D 运动捕捉技术可以用来采集多个演员的动作,这样虚拟(virtual)摄影棚中真实(real)演员的动作就可以用来自动生成逼真的动画(animated)模型并使之具有自然的运动行为。

多个摄像头在多角度下或是单一摄像头在不同的光照条件下都可以精确地获得对材料形状和表面性质进行描述所需的数据,进而自动生成合成的图形模型。这一技术可以用来实现虚拟演员的超写实合成。

3D 捕捉技术的发展,基本可以满足动态捕获人说话时面部表情的需要,从而可以根据语音来合成高度逼真的脸部动画。

针对残障人士的多媒体应用,尤其是对于视力不佳者和老人,也是一个得到广泛关注的研究领域。

“数字时装”的研究目标是开发可以进行交互通信的智能服装,这类服装可以使用无线通信技术来人工地促进人们在社交场合中的交互。这里的创新是将用户的某些想法和感觉广播出去,以便和其他配置有相似设备的用户进行交流。

Georgia Tech 的 Electronic Housecall 系统为病人在其家中提供交互式的健康监护服务,它取决于网络的传输能力,对现有的系统提出了很高的要求。

行为科学中的模型可以用来对人们的交互行为进行建模,从而扩展得到虚拟角色间的自然交互。这类“增强交互”应用可以用来开发真实用户和虚拟用户之间的交互接口,完成诸如让计算机讲故事这种任务。

这些不同的应用领域大大推动了计算机科学的发展,不断刺激新应用的产生,并强烈地吸引着计算机行业的实践者。

## 1.2 多媒体和超媒体

为了将多媒体放置在一个正确的上下文环境中,本节我们将简单回顾多媒体的历史,多媒体历史中最近比较关注的是多媒体和超媒体之间的联结。然后我们介绍用于创建多媒体内容的软件



工具。这将为我们在第2章中深入了解多媒体产品和内容的集成奠定基础。

### 1.2.1 多媒体的历史

使用多媒体作为交流手段的想法可能源于报纸，报纸大量使用文本、图形和图片，是最早的大信息量交流媒介。

运动图片诞生于19世纪30年代，它最初的目的是为了记录那些肉眼无法感知的高速运动。Thomas Alva Edison在1887年发明了可以拍摄运动图片的相机，于是在1910~1927年间无声电影出现了。1927年，《The Jazz Singer》的上映结束了无声电影的时代。

1895年，Guglielmo Marconi在意大利的Pontecchio首次成功地发送了无线电信号。几年后（1901年），无线电信号又成功地横穿了大西洋。无线电最初是用于发送电报的，而现在则成为了语音广播的主要媒介。1909年，Marconi获得诺贝尔物理学奖。（在语音传播的研究领域，来自Quebec的Reginald A. Fessenden几年前早已超越了Marconi，但是并非所有的发明家都得到了应得的荣誉。尽管如此，Fessenden在1928年得到了250万美元作为对其失去的专利的补偿。）

电视是20世纪诞生的全新的传播媒介。它使得视频成为一种广泛应用的媒介，从而改变了大规模传输领域。

而在计算机和多媒体之间建立联系的想法，事实上是不久之前才出现的：

- 1945年 Vannevar Bush (1890—1974) 在其具有里程碑意义的著作[2]中描述了一个名为“Memex”的超媒体系统。Memex是一个具有普遍适用性的个性化存储设备，其中甚至包括了关联链接的概念——它确实可以被看作是万维网的先驱。二战后，大约有6000位在战时为了战争胜利而辛勤工作的科学家发现他们现在有时间 and 精力去考虑一些其他问题，Memex正是这种情况下的产物。
- 1960年 Ted Nelson 开始了Xanadu项目的研究，并发明了术语“超文本”。Xanadu是第一次针对超文本系统的研究尝试，Nelson称超文本系统为“存放文本的神奇场所”。
- 1967年 Nicholas Negroponte 在MIT组建了Architecture Machine研究组。
- 1968年 受Vannevar Bush的《As We May Think》一书的影响和启示，Douglas Engelbart展示了另一个早期的超文本程序On-Line System (NLS)。Engelbart在Stanford研究院的研究小组一直致力于通过计算机技术来提高用户的应用能力，他们所强调的方式是“增强”用户的能力，而非完全的“自动化”。NLS中包含一系列非常重要的概念，如针对思路大纲的编辑器、超文本链接、远程会议、字处理、电子邮件、鼠标定位设备、窗口化软件和帮助系统等[3]。
- 1969年 Brown大学的Nelson和van Dam实现了一个名为FRESS的早期超文本编辑器[4]。今天布朗大学IRIS (Information and Scholarship) 研究院的Intermedia项目正是由这个系统发展而成的。
- 1976年 MIT的Architecture Machine研究组提出了名为“多类媒体”的项目。这导致1978年第一张超媒体视频磁盘——Aspen Movie Map的诞生。
- 1985年 Negroponte和Wiesner共同创建了MIT媒体实验室，该实验室成为在数字视频和多媒体领域具有主导地位的研究机构。
- 1989年 Tim Berners-Lee向欧洲核技术研究理事会(CERN)提出了World Wide Web的概念。
- 1990年 Kristina Hooper Woolsey开始领导Apple的多媒体实验室。该实验室拥有100多位员工，并以教育方面的应用为主要研究目标。

- 1991 年 MPEG-1 成为数字视频的国际标准，之后在此基础上开发了一系列更新的标准，如 MPEG-2、MPEG-4 等。
- 1991 年 PDA 的诞生开启了计算机应用的一个新的时代，对多媒体而言更是如此。随着 1996 年无键盘 PDA 的市场化，这一发展趋势得到了进一步延续。
- 1992 年 JPEG 成为数字图像压缩的国际标准。它的进一步发展导致了 JPEG 2000 标准的诞生。
- 1992 年 产生第一个网络上的 Mbone 音频多播。
- 1993 年 Illinois 大学的 Supercomputing Applications 国家中心开发了 NCSA Mosaic，这是第一个比较成熟的浏览器，从而开创了 Internet 信息访问的新时代。
- 1994 年 Jim Clark 和 Marc Andreessen 开发了 Netscape 浏览器。
- 1995 年 Java 语言诞生。Java 语言可以用来开发与平台无关的应用程序。
- 1996 年 DVD 技术的产生使得一整部高清晰度的电影可以被收录在一张磁盘上。人们预言 DVD 格式将改变整个音乐、游戏和计算机行业。
- 1998 年 W3C 发布了 XML 1.0。
- 1998 年 具有 32MB 闪存的手持 MP3 设备成为市场上深受消费者青睐的产品。
- 2000 年 World Wide Web (WWW) 预计已有 10 亿张网页的规模。

### 1.2.2 超媒体和多媒体

Ted Nelson 在 1965 年提出了“超文本”这个术语。我们通常把一本书看做是线性媒体，要从头到尾顺序阅读，而浏览超文本系统就是非线性的，可以利用指向文档中其他部分或是其他文档的链接来进行。图 1-1 说明了这种关系。

超媒体 (hypermedia) 并不一定是基于文本的，它同样可以包含其他媒体，如图形、图像，特别是具有连续性的媒体，如声音和视频。很明显，Ted Nelson 也是第一个使用这个术语的人。World Wide Web (WWW) 是超媒体应用的一个最好的例子。

正如我们所看到的那样，多媒体 (multimedia) 意味着除了应用传统媒体 (如文本和图形) 之外，计算机信息还可以通过音频、图形、图像、视频和动画等多种形式来表达。可以将超媒体看做是多媒体应用的一个特例。

典型的多媒体应用包括数字视频的编辑和制作系统、电子报纸和杂志、WWW、游戏、在线参考咨询工作 (如百科全书)、游戏、群件、家庭购物、交互电视、多媒体课件、视频会议、视频点播和交互电影等。

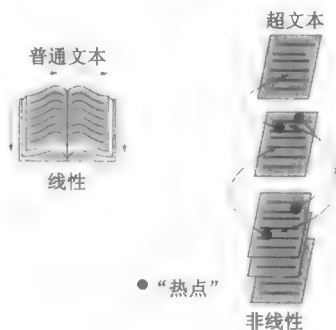


图 1-1 非线性的超文本

## 1.3 World Wide Web

World Wide Web (万维网，简称 WWW) 是目前规模最大也是最常用的超媒体应用。WWW 之所以如此流行，是因为用户可以迅速地从 Web 服务器获取海量的信息，方便地发布信息，并快捷有效地在 Web 浏览器上访问这些信息。WWW 技术是由 World Wide Web Consortium (W3C，万维网协会) 来负责维护和开发的，而 Internet Engineering Task Force (IETF) 则负责对该技术进行标准化。W3C 列出了 WWW 的三个目标：对网络资源的普遍访问 (任何人在任何地点都可以访问)，对信息的有效浏览和对已发布内容的可靠使用。

### 1.3.1 WWW 的历史

令人惊讶的是，目前最为流行的网络多媒体应用竟然可以追根溯源到核物理！正如在前面章

节中所提到的那样, Tim Berners-Lee 向 CERN(European Center for Nuclear Research)提出将 World Wide Web 技术作为一种组织和共享其工作与实验结果的方式。下面简要回顾一下 WWW 创建的历程:

- 20 世纪 60 年代 人们认识到文档的格式需要具有良好的可读性, 并能对结构和元素进行区分。Charles Goldfarb、Edward Mosher 和 Raymond Lorie 为 IBM 开发了通用标记语言 (Generalized Markup Language, GML)。
- 1986 年 ISO 发布了标准通用标记语言 (SGML) 的最终版本, 它主要是基于更早的 GML。
- 1990 年 在 CERN 的支持下, Tim Berners-Lee 开始在 NeXTStep 工作站上开发超文本服务器、浏览器和编辑器。为此, 他发明了超文本标记语言 (HyperText Markup Language, HTML) 和超文本传输协议 (HyperText Transfer Protocol, HTTP)。
- 1993 NCSA 发布其 X Windows 系统上基于 Marc Andreessen 的 Mosaic 的  $\alpha$  版。这是第一个流行的浏览器, 微软的 Internet Explorer 就是基于 Mosaic 的。
- 1994 年 Marc Andreessen 和他在 NCSA 的一些同事与 Dr. James H. Clark (他也是 Silicon Graphics Inc. 的创建者) 共同组建了 Mosaic 通信公司。11 月, 公司更名为 Netscape 通信公司。
- 1998 年 W3C 接受 XML 1.0 规范作为其发布的推荐标准。XML 从此取代 HTML 成为 W3C 关注的主要焦点。

### 1.3.2 HTTP

HTTP 最初是为了传输超媒体而设计的, 但它也可以用来传输其他类型的文件。HTTP 是一种“无状态”的请求/响应协议, 也就是说, 客户端在建立和 HTTP 服务器的连接后, 发送请求信息, 并在服务器响应后中断连接, 本次连接和下一次请求之间没有任何关系, 一次请求过程对应一个 HTTP 连接。

基本的请求格式如下所示:

```
Method URI Version
Additional-Headers

Message-body
```

统一资源标识符 (URI) 指出被访问的资源, 例如以 “http://” 开头的主机名。URI 也可以是同一资源定位符 (URL)。这里, URI 还可以包括查询字符串 (用于需要提交数据的某些交互动作)。Method 是在 URI 上交换信息和执行操作的方式, 主要应用 GET 和 POST 两种方法。GET 表明请求的信息包含在请求字符串内, 而 POST 则表明 URI 所指向的资源需要去考虑消息体中的信息。POST 通常用于提交 HTML 表单。Additional-Headers 指定客户端的附加参数。例如, 当请求访问本书的网站时, 可能会产生下面的 HTTP 消息:

```
GET http://www.cs.sfu.ca/mmbook/ HTTP/1.1
```

基本的响应格式为

```
Version Status-Code Status-Phrase
Additional-Headers

Message-body
```

Status-Code 表示响应类型 (或错误类型) 的编号, 而 Status-Phrase 则是相应的文本描述。当请求被成功处理时, 这两者分别是 200 和 OK; 当 URI 不存在时, 则分别为 404 和 Not Found。以上是两种最为常见的 Status-Code 和 Status-Phrase。例如, 在响应对本书网站的访问请

求时，Web 服务器可能会返回以下信息：

```
HTTP/1.1 200 OK Server:
[No-plugins-here-please] Date: Wed, 25 July 2002
20:04:30 GMT
Content-Length: 1045 Content-Type: text/html

<HTML>
...
</HTML>
```

### 1.3.3 HTML

HTML 是用来在 World Wide Web 上发布超媒体信息的语言。它的定义使用了 SGML 规范，并派生出了一组用来描述通用文档结构和格式的元素。由于 HTML 使用 ASCII 码，因此它可以移植到任何（甚至是非二进制兼容的）计算机硬件上，这一特性使得全球信息交换成为可能。HTML 目前的版本为 1999 年制定的 4.01 版本。下一代 HTML 语言将是 XHTML，它是将 HTML 和 XML 相结合的产物。

HTML 使用标记来描述文档元素。标记使用类似于<token params>的格式来定义文档元素的起始点，用类似于</token>的格式来定义元素的结束点。某些元素只有内联参数，所以不需要结束标记。HTML 将文档分为 HEAD 和 BODY 两个部分，形式如下：

```
<HTML>
<HEAD>
...
</HEAD>
<BODY>
...
</BODY>
</HTML>
```

HEAD 部分描述文档的定义，这个部分将在文档显示前被解析。这个部分包括页面标题、资源链接以及作者定义的元信息。BODY 部分描述文档的结构和内容。常用的结构元素包括段落、表、表单、链接、链表和按钮等。

10

下面是一个 HTML 页面的简单例子：

```
<HTML>
<HEAD>
  <TITLE>
    A sample web page.
  </TITLE>
  <META NAME = "Author" CONTENT = "Cranky Professor">
</HEAD> <BODY>
  <P>
    We can put any text we like here, since this is
    a paragraph element.
  </P>
</BODY>
</HTML>
```

HTML 还有其他更为复杂的结构并可以和其他标准混合使用。HTML 规范经过不断发展，现在已经支持和脚本语言的集成，在客户端对元素和属性进行动态操纵（动态 HTML），以及使用名为级联样式表（Cascading Style Sheet, CSS）的标记语言来对显示参数进行模块化的定制。当然，HTML 具有严格的、非描述性的结构元素，也很难达到模块化。

### 1.3.4 XML

对于 WWW 的标记语言而言，数据、结构和视图的模块化特性是很有必要的。我们希望用户

或应用能够自己定义文档中的标记（结构）以及它们之间的关系，并在 XML 文件中使用这些标记来定义数据，最后在另一个文档中定义如何显示这些标记。

假设你希望根据用户的查询请求从数据库中检索股票信息。使用 XML 语言，你需要事先为股票数据创建全局文档类型定义（DTD）。这样，服务器端的脚本程序就可以根据 DTD 定义的规则，利用数据库中的数据来生成满足查询条件的 XML 文档。最后，根据显示设备的不同，用户将会收到 XML 样式表，以便在不同显示设备（例如 CRT 显示器或是手机屏幕）上都能得到最佳的视觉效果。

XML 目前的版本是 1998 年 2 月通过的 1.0 版本。XML 的语法和 HTML 相似，但 XML 更为严格。所有的标记都必须小写，如果一个标记只有内联数据，那么它也必须包含结束符，例如 `<token params />`。XML 还使用名称空间，以便区分不同 DTD 中具有相同名字的标记。我们也可以通过 URI 来导入 DTD。下面是一个 XHTML 文档的定义，我们可以看一下 XML 的文档结构：

```
11 <?xml version="1.0" encoding="iso-8859-1"?>
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transition.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
        ... [html that follows
            the above mentioned
            XML rules]
    </html>
```

所有的 XML 文档都以 `<?xml version="ver"?>` 开头。`<!DOCTYPE...>` 是用来导入 DTD 的特殊标记。由于它实际上是 DTD 的定义，因此并不遵循 XML 规则。`xmlns` 为文档元素定义了唯一的名称空间。在上面的例子中，名称空间是 XHTML 规范的说明网页。

以下是其他一些和 XML 相关的规范：

- **XML 协议** 用来在进程间交换 XML 信息。它被用来替代 HTTP 协议，并将进一步扩展以支持网络上的进程间通信。
- **XML Schema** 一种结构化和功能更加强大的语言，用来定义 XML 数据类型（标记）。和 DTD 不同，XML Schema 用 XML 标记来进行类型定义。
- **XSL** 基本来说，这是 XML 的 CSS。另一方面，XSL 更为复杂，它由三个部分构成：XSLT（XSL Transformations）、XPath（XML Path Language）和 XSL 格式对象。
- **SMIL** 同步多媒体集成语言，读作“smile”。它是 XML 应用（使用预定义的 DTD）的一个特例，允许我们采用脚本的形式来定义任意媒体类型和用户输入之间的交互。例如，它可以用来实现流媒体和幻灯片之间的同步显示，用户对这两者的随机浏览都将得到响应。

### 1.3.5 SMIL

正如我们需要 HTML 作为发布文本-文档的可读标记语言那样，我们也需要一种发布多媒体表现的标记语言。多媒体表现具有一些独有的特点：在文本文档中，所有的文本信息是顺序阅读的，并且同时显示出来，而多媒体表现则具有很多内容随时间变化的元素，如视频和音频等。所以，多媒体标记语言必须支持对不同多媒体元素的调度和同步，并定义这些元素和用户之间的交互。

W3C 在 1997 年成立了一个工作组进行多媒体同步语言规范的制定。该工作组在 1998 年 6 月制定了后来成为 W3C 推荐标准的 SMIL 1.0 规范。随着 HTML 在 XML 中重新定义（XHTML 规范），SMIL 1.0 也进行了修订，在功能上有所增强。SMIL 2.0 规范集成了 HTML，它于 2001 年 8 月被 W3C 接受为推荐标准。

和 XHTML 类似，SMIL 2.0 在 XML 中也是使用模块化的方法定义的。所有的 SMIL 元素都可以被归入到不同的模块中——模块是功能类似的元素、属性和值的集合。在模块化领域，应用

程序无需包含所有可用的模块。因此便有了语言规范（profile）的概念，它定义了一组遵循一定集成规则的特定模块。SMIL 2.0 有一个包括几乎所有 SMIL 模块的核心语言规范，一个只包含支持基本功能所需模块的基本规范，以及用来集成 SMIL 和 HTML 的 XHTML+SMIL 规范。后者包括了大部分的 XHTML 模块和 SMIL 的定时模块（这里没有包括结构模块，因为 XHTML 有自己的结构模块）。

SMIL 的语言结构和 XHTML 类似。根元素为 smil，它包含两个元素 head 和 body。head 包含除同步之外的信息——元信息、布局信息和内容控制（例如媒体的码率）。body 则包含和显示哪些资源以及何时显示这些资源相关的信息。

SMIL 中有三类资源同步的方法：seq、par 和 excl。seq 规定相关的元素必须按照事先定义的顺序进行播放（顺序）。相应的，par 规定相关元素同时被播放（并行）。excl 表示同一时刻只能有一个元素被播放（互斥），但是事先并不规定播放顺序。

下面是 SMIL 代码的一个例子：

```
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0"
"http://www.w3.org/2001/SMIL20/SMIL20.dtd">
<smil xmlns=
"http://www.w3.org/2001/SMIL20/Language">
<head>
  <meta name="Author" content="Some Professor" />
</head>
<body>
  <par id="MakingOfABook">
    <seq>
      <video src="authorview.mpg" />
      
    </seq>

    <audio src="authorview.wav" />
    <text src="http://www.cs.sfu.ca/mmbook/" />
  </par>
</body>
</smil>
```

13

SMIL 文档可以选择是否使用<!DOCTYPE...>来指示导入 SMIL DTD。如果导入了 DTD，那么编译器就会根据 DTD 来对文档进行验证。SMIL 文档以<smil>开头并用 xmlns 属性来指定默认的名称空间。<head>部分指定文档的作者，而<body>部分包含同步信息和需要显示的资源。

在上面的例子中，名为“authorview.mpg”的视频资源，名为“authorview.wav”的音频资源和位于“http://booksite.html”的 HTML 文档将在开始时同时播放。当视频结束时，则显示图片“onagooodday.jpg”，此时音频和 HTML 文档则仍在播放中。此时，在音频文件中作者将表示对观众的感谢并结束这次访谈。

关于 SMIL 的其他信息以及可用的模块可以在 W3C 的网站上获取。

## 1.4 多媒体软件工具概述

在本节中，我们将简单介绍用于多媒体处理的软件工具。了解这些软件工具仅仅是一个开始，完成一个功能全面的多媒体项目不但需要非常出色的编程技巧，还要使用已有工具发挥网络和计算机的强大功能<sup>①</sup>。

下面是我们将要介绍的几类软件。

① 可以从相关的网站上了解软件工具的用途。在常见的多媒体课程中，第一项作业往往就是使用本节介绍的工具制作一个简单的多媒体产品。有些工具功能强大，可以用来作为课程设计的一部分。

- 音序和记谱<sup>①</sup>
- 数字音频
- 图形和图像编辑
- 视频编辑
- 动画
- 多媒体编著

#### 1.4.1 编曲和谱曲

- **Cakewalk** Cakewalk 是 Pro Audio 曾经使用的名称。开发这款音序和编辑软件的公司是 Twelve Tone Systems, 该公司也在 Internet 上以较低的价格出售软件的简略版本“Cakewalk Express”。
- **术语音序器 (sequencer)** 来源于在 MIDI 音乐语言 (MIDI 中的事件, 参见 6.2 节) 中存储音符序列的老式设备。我们也可以向乐曲中插入 WAV 文件和 Windows MCI 命令 (用于动画和视频)。这里所说的 MCI 是 Windows API 中一个常见的组件。
- **Cubase** Cubase 是另一款音序/编辑程序, 具有和 Cakewalk 类似的功能。它包括了一些数字音频的编辑工具 (见下面的介绍)。
- **Macromedia Soundedit** Soundedit 是一款用来为多媒体项目和网页制作音频的软件。它已经开发得相当成熟, 并能和其他的 Macromedia 产品集成使用, 如 Flash、Director 等。

14

#### 1.4.2 数字音频

数字音频工具主要用来访问和编辑构成音频的真实采样的声音。

- **Cool Edit** Cool Edit 是一款非常流行的功能强大的数字音频工具集, 具有可以和专业音频工作室相媲美的处理能力 (对于 PC 用户而言), 包括多声道的生成、声音文件编辑和数字信号处理。
- **Sound Forge** Sound Forge 是一款基于 PC 的高级程序, 用来编辑 WAV 文件。它可以通过声卡从光驱、磁带或是麦克风采集声音, 以供进一步混音和编辑。它还支持添加特殊音效。
- **Pro Tools** Pro Tools 是一款运行在 Macintosh 或 Windows 平台上的高端集成音频产品和编辑环境。Pro Tools 提供了便捷的 MIDI 制作和操作功能, 以及强大的音频混合、录制和编辑软件。

#### 1.4.3 图形和图像编辑

- **Adobe Illustrator** Illustrator 是一款功能强大的用于制作和编辑向量图的发布工具, 可方便地导出向量图以便在 Web 上使用。
- **Adobe Photoshop** Photoshop 是图形图像处理和制作的标准工具。图形、图像和文本可以分在不同层次进行独立的操作, 非常灵活。此外, 它的“滤镜工厂” (filter factory) 可以实现非常复杂的光学效果。
- **Macromedia Fireworks** Fireworks 是专门用来制作网页图形的软件。它包括位图编辑器、向量图编辑器以及用于按钮和翻转的 JavaScript 生成器。
- **Macromedia Freehand** Freehand 是一个文本和网页图形编辑工具, 它支持多种位图格式, 如 GIF、PNG 和 JPEG。这些都是基于像素的格式, 这种格式指定了每个像素。它同样支

① 音序软件主要用于音乐制作, 能录下合成器所发出的 MIDI 信息, 可用来编曲、进行音乐创作; 记谱软件一般只具有很少的音序功能, 而主要用于记录乐谱, 用来出版音乐书籍或乐谱。

持基于向量的格式,这种格式只需指定线段的两个端点,而不必指定每个像素,例如 SWF (Macromedia Flash) 和 FHC (Shockwave Freehand)。它还可以读入 Photoshop 的格式。

#### 1.4.4 视频编辑

- **Adobe Premiere** Premiere 是一款简单直观的非线性视频编辑工具——可以将视频片段以任意次序放置。视频和音频排列在不同的轨道上,就好像乐谱那样。它提供了大量的音频和视频轨道、叠加和虚拟片段。对于片段,它有一个包含内置的转换、过滤器和运动的库,以便高效开发多媒体产品。
- **Adobe After Effects** After Effects 是一款功能强大的视频编辑工具,支持用户给已有的视频文件添加特殊效果或对已有视频文件进行修改,如光照、阴影和运动模糊等。和 Photoshop 类似,它也是用图层来进行对象的独立编辑。
- **Final Cut Pro** Final Cut Pro 是 Apple 为 Macintosh 平台提供的视频编辑工具。它可以从大量数据源中(如电影和 DV)采集音频和视频数据。它提供了一套非常完整的环境,从视频的采集到编辑、色彩修正,以及最终将结果输出到视频文件或是在网络上广播。

15

#### 1.4.5 动画

##### 1. 多媒体 API

Java 3D 是 Java 用来构建和显示 3D 图形的 API,和 Java Media Framework 处理媒体文件的方式类似。它提供了一套基本的对象原语(立方体、曲线等)来让开发人员进行场景的构建。由于它是在建立在 OpenGL 或 DirectX(用户可以选择)之上的抽象层,因此可以支持图形加速。

DirectX 是一个支持视频、图像、音频和 3D 动画的 Windows API,是目前 Windows 多媒体应用程序(如计算机游戏)开发中应用最广泛的 API。

OpenGL 诞生于 1992 年,是目前最为流行的 3D API。OpenGL 具有高度的可移植性,可以运行在目前所有流行的操作系统上,如 UNIX、Linux、Windows 和 Macintosh。

##### 2. 显示工具

3D Studio Max 包括一组高端的专业工具,用于完成人物动画、游戏开发和视觉效果的制作。使用这一工具建立的模型在很多游戏中得到应用,如 Sony Playstation。

Softimage XSI(曾被称为 Softimage 3D)是一款功能强大的建模、动画和显示软件包,用于在游戏和电影中制作动画和生成特殊效果。

Maya 是 Softimage 的竞争对手,它包括了一个完整的建模软件包,拥有多种不同的建模和动画工具,例如构造逼真的衣物和皮毛的工具。

RenderMan 是 Pixar 开发的显示工具包,在构建复杂的图形外观和图像方面有着出色的性能,并在很多电影中得到应用,如 *Monsters Inc.* 和 *Final Fantasy: The Spirits Within* 等。它还可以从 Maya 中导入模型。

GIF Animation Package 为了能在 Web 应用中对小型动画进行简单有效的开发,很多共享软件和其他程序都支持 GIF 动画图像的制作。GIF 包含了多幅图像,并通过它们之间的循环构成简单的动画。Gifcon 和 GifBuilder 就是两个这样的程序。Linux 也提供了一些简单的动画工具,如 animate。

16



### 1.4.6 多媒体编著

能够提供包括交互式用户控制在内的一整套完整的多媒体表现能力的工具，称为编著（authoring）程序。

- **Macromedia Flash** Flash 通过一种类似于乐谱的方法来支持交互式电影的创作，因为并行的事件序列排列在时间线上，就好像乐谱中的音符一样。电影中的元素在 Flash 中称为符号（symbol）。符号被添加到称为库的中心存储库中，并可以加入到电影的时间线上。在某一时刻显示这些符号时，它们就出现在舞台（Stage）上。舞台给出了电影在某一时刻的内容，并可以通过 Flash 内置的工具进行操作和移动。Flash 电影通常用来在 Web 上显示电影或游戏。
- **Macromedia Director** Director 用一种类似于电影的方法进行交互式表现的创作。这个功能强大的程序包括一种内置脚本语言 Lingo，可以进行复杂交互电影的制作<sup>①</sup>。Director 中的角色包括位图分镜、脚本、音乐、声音和调色板。Director 可以读入多种不同的位图格式。程序对交互性有着良好的支持，Lingo（具有自己的调试器）则允许更多的控制行为，包括对外部设备的控制，如对 VCR 和视频唱片播放器。Director 还有网页编著的能力，可以进行 Web 上 Shockwave 电影的制作。
- **Authorware** Authorware 是一个成熟的并得到广泛支持的编著工具，由于它采用基于流程图（即所谓的图标/流控制模式）的思想，因此对计算机学生来说易于学习。它可以为文本、数字电影、图形和声音添加超链接。它还提供用 PC、Mac 不同版本创建的文件之间的兼容性。Shockwave Authorware 应用可以整合 Shockwave 文件，包括 Director 电影、Flash 动画和音频。
- **Quest** Quest 和 Authorware 在很多方面都具有相似性，它使用类似于流程图的方法。但是，流程图中的节点可以用更为抽象的方式（称为帧）来封装信息。因此，图标之间的连接更为概念化，并不总是表示程序中控制的流向。

## 1.5 进一步探索

Steinmetz 和 Nahrstedt[5]中的第 1 章和第 2 章对多媒体的概念的总体介绍很有参考价值。

本书的配套网站与相关领域的最新发展保持着很好的同步。网站中 Further Exploration 目录下的第 1 章提供了不少关于多媒体历史的链接。作为开头，网站提供了 Vannevar Bush 关于 Memex 系统概念的文章。这些文章在过去和现在都被认为极具启发性。尽管写于 50 年前，但它还是预言到了很多当前的发展，包括传真机和关联记忆模型。Nielsen 的书[6]则从总体上介绍了超文本和超媒体。如果希望了解更新的信息，Jeffay 和 Zhang[1]的研究论文提供了更为深入的背景介绍和未来研究的指导。

网站中其他链接的包括了以下方面的信息：

- Ted Nelson 和 Xanadu 项目。
- Nicholas Negroponte 在 MIT Media Lab 的工作成果。Negroponte 关于多媒体的著作[7]是一个经常被引用的经典作品。
- Douglas Engelbart 和 “On-Line System” 的历史。
- MIT Media Lab。Negroponte 和 Wiesner 共同创建了 MIT Media Lab，它仍在不断壮大中，

① 所以，对于多媒体的课程设计而言，Director 是一个不错的选择，因为它具有强大的开发能力，而且无需使用令人头疼的 C++。

并被认为是世界上最有影响力的学术理念的发源地。

- 客户端执行。Java 和客户端执行的提出始于 1995 年,“Duke”(第一个 Java applet)同样可以在该书的网站上找到链接。

Buford 的书[8]的第 12 章对编著进行了详细介绍。Neuschotz 的文章[9]给出了创建简单的基于 Lingo 的交互式 Director 电影的步骤。

其他的链接包括:

- 数字音频。这个网页包括了指向 Sonic Foundry 公司的链接,从中可以找到 Sound Forge 的信息,一个 Sound Forge 的范例程序,以及生成的 WAV 文件。这个范例采用一种比较复杂的方式来混合左右声道的信息。这个工具可以非常便捷地创建复杂的特殊效果。Digidesign 是一家提供高端 Macintosh 软件的公司,还销售具有专门处理功能的电路板。
- 音序和记谱。
- 图形和图像的编辑信息。
- 视频编辑产品和信息。
- 动画网址。
- 多媒体编著工具。
- XML。

## 1.6 练习

1. 说出三种较有新意的 Internet 应用或多媒体应用。说出你认为它们有新意的理由。
2. 用自己的话简单解释 Memex 以及它在超文本方面的作用。我们今天还能继续 Memex 应用吗?在你自己的工作中,你如何在实际工作中应用 Memex 的理念?
3. 假设你需要在 Internet 上传输气味。现在我们在某处有一个气味传感器,并且希望将芳香向量(以此为例)传输到一个接收器以复制相同的气味。试设计一个这样的系统。列出需要考虑的三个主要问题和这类传输系统的两个应用。提示:考虑医学应用。
4. 人物或物体的跟踪可以通过视觉或声音手段来完成。视觉系统的准确度较高,但是代价相对较为昂贵;而使用一组麦克风就可以在付出较少的费用的情况下对人的方位进行精确度要求不高的定位。因此,视觉和声音方法的融合是很有意义的。上网查找是否有人应用这一理念开发用于视频会议系统的工具。
5. 非照片逼真度的图形表示那些并非用来构建照片图像的计算机图形。一个例子是会议系统(让我们再一次关注这个最前沿的应用)。例如,如果我们跟踪嘴唇的运动,我们可以生成和我们脸部相应的动画。如果我们不希望使用自己的脸部,我们可以用其他的脸部来代替——脸部特征模型可以将正确的嘴唇动作匹配到另外一个模型上。试查找谁在进行 avatar 生成的研究(avatar 是会议参与者身体动作的模拟表示)。
6. 水印是在数据中嵌入隐藏信息的技术。它具有法律的内涵:这幅图像是否被抄袭?这幅图像是否被篡改?这是由谁、在哪里完成的?考虑这类秘密嵌入在图像中、并且能够被辨识的信息,并回答上面这些问题(类似的问题来自移动电话的使用,我们可以用什么来确定是谁、在哪儿、在什么时候用这个手机?)。

18

## 1.7 参考文献

- [1] K. Jeffay and H. Zhang, *Readings in Multimedia Computing and Networking*, San Francisco: Morgan Kaufmann, CA, 2002.
- [2] Vannevar Bush, “As We May Think,” *The Atlantic Monthly*, Jul. 1945.
- [3] D. Engelbart and H. Lehtman, “Working Together,” *BYTE Magazine*, Dec. 1998, 245–252.

- [4] N. Yankelovitch, N. Meyrowitz, and A. van Dam, "Reading and Writing the Electronic Book," in *Hypermedia and Literary Studies*, ed. P. Delany and G.P. Landow, Cambridge, MA: MIT Press, 1991.
- [5] R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications and Applications*, Upper Saddle River, NJ: Prentice Hall PTR, 1995.
- [6] J. Nielsen, *Multimedia and Hypertext: The Internet and Beyond*, San Diego: AP Professional, 1995.
- [7] N. Negroponte, *Being Digital*, New York: Vintage Books, 1995.
- [8] J.F.K. Buford, *Multimedia Systems*, Reading, MA: Addison Wesley, 1994.
- [9] N. Neuschotz, *Introduction to Director and Lingo: Multimedia and Internet Applications*, Upper Saddle River, NJ: Prentice Hall, 2000.

## 第2章 多媒体编著和工具

### 2.1 多媒体编著

多媒体编著就是多媒体作品（有时又被称为“电影”或“展现”）的制作。由于我们要从计算机科学的角度来探讨这个问题，因此我们更关心交互式的应用程序。而且，我们需要考虑静态图像编辑器（如 Adobe Photoshop）以及简单的视频编辑器（如 Premiere），因为这类应用程序帮助我们建立交互式的多媒体项目。

交互的程度取决于应用本身，其范围从没有任何交互（如幻灯片的演示）到完全沉浸式的虚拟现实。

在幻灯片的演示中，交互主要包括对演示速度的控制（如单击以转至下一页）。下一个层次的交互是对播放顺序的控制和下一页的选择。再下一个层次是对媒体的控制：视频的开始/结束、文本搜索、视图的滚动和缩放。如果我们可以实现对变量（例如对数据库的查询请求）的控制，那么可以实现更多的控制。

如果我们可以对对象进行控制，例如在屏幕上移动对象和进行交互式的游戏等等，那么控制的水平就可以得到本质的提高。最后，我们可以对一个完整的仿真过程进行控制：我们在场景中移动视角，控制场景对象。

很长一段时间以来，人们都在考虑多媒体项目中应该包含什么内容，相关的参考将在本章最后给出。

在本节中，我们将介绍如下内容：

- 多媒体编著的模式
- 多媒体作品
- 多媒体展现
- 自动编著

介绍自动编著主要是为了处理常见的编著问题以及利用自动工具（例如人工智能技术的使用）支持编著工作。首先，我们考虑为遗留文档执行自动链接的程序。

在对多媒体的范例进行介绍后，我们将介绍一些实用的多媒体内容生成工具——构成一个完整的多媒体作品库的软件工具。这里我们将详细介绍目前得到广泛应用的一些标准程序。

20

#### 2.1.1 多媒体编著的模式

编著是创建多媒体应用的过程。大多数编著程序都使用某种常见的编著模式，这些编著模式有时也称为编著范例，编著范例将简化对创建多媒体应用所使用方法的理解。

常见的编著模式如下所示：

- 脚本语言模式

这种模式的思想是使用一种专门的语言来实现交互（按钮、鼠标等）并支持条件、跳转、循环、函数/宏等。一个例子是 Asymetrix Learning Systems 的 Toolbook 程序中的 OpenScript 语言。OpenScript 类似于标准的面向对象和事件驱动的程序设计语言。下面的例子显示了一个 Toolbook 小程序。由于必须对对象库进行学习，因此这样的语言都有一条学习曲线，当然，所有的编著工具都是如此，即使那些使用标准 C 语言作为脚本语言的工具也是如此。

```

-- load an MPEG file
extFileName of MediaPlayer "theMpegPath" =
    "c:\windows\media\home33.mpg";
-- play
extPlayCount of MediaPlayer "theMpegPath" = 1;
-- put the MediaPlayer in frames mode (not time mode)
extDisplayMode of MediaPlayer "theMpegPath" = 1;
-- if want to start and end at specific frames:
extSelectionStart of MediaPlayer "theMpegPath" = 103;
extSelectionEnd of MediaPlayer "theMpegPath" = 1997;
-- start playback
get extPlay() of MediaPlayer "theMpegPath";

```

#### • 幻灯片显示模式

默认的幻灯片显示是一种线性显示过程。尽管有执行页面间跳转的工具，但却很少有人使用这一功能。**PowerPoint** 和 **ImageQ** 就是这样的范例。

#### • 层次模式

这里，用户可控的元素以树状结构组织在一起。这种模式通常用在菜单驱动的程序中。

#### • 图标/流程控制模式

工具箱中包含有图形化的图标，而编著过程可以通过创建带有图标的流程图来实现。这类模式的一个标准例子是 **Macromedia** 的 **Authorware**。图 2-1 显示了一个流程图。我们可以使用 Map（即一个子例程）图标来对元素分组，也可使用简单的流程图元素，如 IF 语句、CASE 语句等来对元素分组。这样，我们可以很方便地实现简单的动画。

#### • 框架模式

和图标/流程控制模式相似，工具箱中包含有图形化的图标，而编著过程可以通过创建带有图标的流程图来实现。然而在这里，图标间的链接更为概念化，而不代表程序的真正流向。这类模式的一个例子是 **Allen Communication** 的 **Quest**。流程图中包含了由“框架”构成的“模块”。框架则由对象构成，例如文本、图形、音频、动画和视频，这些对象都可以响应事件。此外，这里使用的脚本语言是我们常用的 C 语言。图 2-2 显示了一个 **Quest** 框架。

#### • 卡片/脚本模式

这种模式使用简单的索引-卡片结构来开发多媒体作品。由于我们可以使用链接，因此这是一条开发使用超文本或超媒体应用的捷径。**Apple** 的 **HyperCard** 是最早使用这类模式的，另一个例子是 **Knowledge Adventure** 的 **HyperStudio**。后者目前在很多学校里得到应用。图 2-3 显示了 **HyperStudio** 栈中的两张卡片。

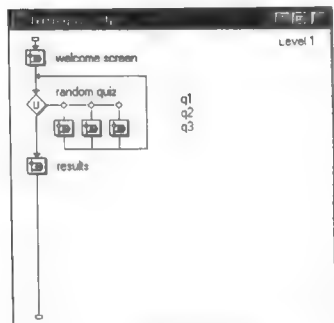


图 2-1 Authorware 流程图

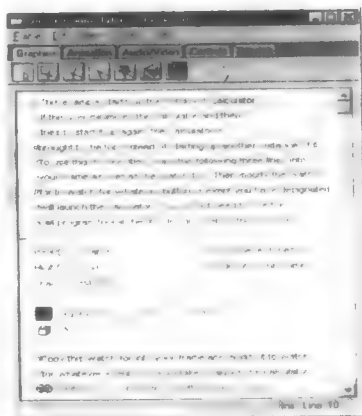


图 2-2 Quest 框架

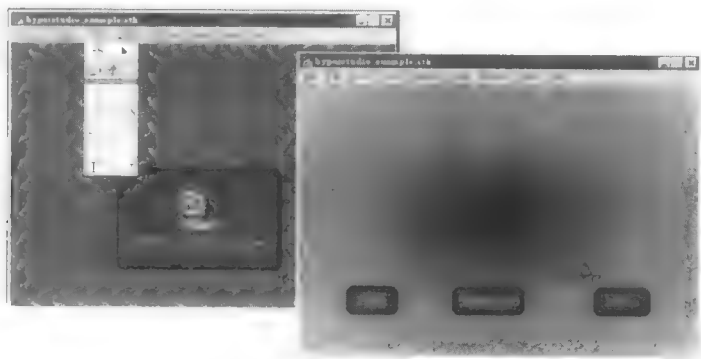


图 2-3 HyperStudio 栈中的两张卡片

### • 角色/乐谱/脚本模式

在这种模式中，横轴是时间轴，用电子表格的形式水平显示，纵轴又叫轨道，用以显示多媒体作品中实例化的不同角色。由于这些轨道控制同步行为，因此这种模式和乐谱非常相似。多媒体元素类似于角色的“演员表”，而“脚本”则是基本的事件过程或由定时器事件引发的过程。通常，你可以编写自己的脚本。所以，这和传统的“脚本语言”的概念非常相似，传统的脚本语言非常简洁，抽象程度较低，因为这只是某种专门用途的脚本。

Macromedia 的 Director 是这种模式的重要例子。Director 使用 Lingo 脚本语言，它是一种面向对象并且由事件驱动的语言。

## 2.1.2 多媒体作品

多媒体项目往往需要一组具有专业技能的人员参与。在本书中，我们更加专注技术层面，但是多媒体作品的制作通常不仅需要程序员，还需要艺术导演、图形设计师、艺术指导、制片人、项目经理、编剧、用户界面设计师、音效师、摄影师以及 3D 与 2D 动画制作人员的共同参与。

在完成作品的 40%前，此时主要的工作一般只涉及程序员；当完成 65%~70%时，应该能够有作品的 alpha 版本（早期版本，不包括所有计划内的功能和特征）。一般来说，设计过程由情节串联、构造流程图、设计原型系统、进行用户测试以及并行的媒体宣传等阶段构成。编程和调试阶段将与市场营销互相协调进行，然后便是产品的分销阶段。

情节串联板（又称故事板）通过一系列的草图描述了多媒体概念的基本设想。这和视频中的关键帧类似，故事将围绕这些“停止点”展开。流程图通过插入导航信息（多媒体概念的结构和用户交互）来对故事板进行组织。安排导航信息最为可靠的方法是选用传统的数据结构。层次化的系统是一种最为简单的组织策略。

多媒体不同于其他展现，必须十分仔细地考虑如何在新产品中组织不同“房间”之间的运动。例如，我们正在导航一次非洲之旅，但我们还需要带些样品回到博物馆以便仔细观察，那么我们如何才能有效地从一个场景转到另一个场景？流程图有助于设计出解决方案。

流程图阶段之后是详细功能规范的开发。这是指对展现中每个场景进行逐帧的走查，包括所有的屏幕动作和用户交互。例如，当鼠标掠过某个角色时，它会做出反应；或是当用户单击鼠标时，角色会做出某种动作。

构建原型和测试是设计阶段的最后一个组成部分。一些多媒体设计者在这个阶段已经开始使用编著工具，尽管这个中间的原型并不会用到最终的作品中或在另一个工具中使用。用户测试是最终的开发阶段前的一个极为重要的步骤。

### 2.1.3 多媒体展现

在本节中，我们将简要介绍影响多媒体内容展现效果的一些重要因素以及内容设计的指导方针。

#### 1. 图形风格

我们需要仔细考虑展现中的配色方案和文字的视觉效果。很多的展现都是针对商用显示器，而非屏幕。此外，还需要考虑人类视觉的动态性以决定展现的尺寸。这里的大多数观点都参考自 Vetter 等人的著作[2]，如图 2-4 所示。

#### 2. 色彩的原则和方针

某些配色方案和艺术风格与某种主题、风格的匹配效果最好。例如，对于室外场景，配色方案可以较为自然和鲜艳；对于室内场景则应该朴素一些。可以采用的艺术风格包括油画、水彩画、彩色铅笔画或是粉笔画。

通常建议不要用太多的颜色，因为会使人分心，这与用色一致——颜色可以用来表示主题的改变。

#### 3. 字体

为了提高视觉的效果，大字体（18~36 磅）是最佳的选择，每屏的行数不应多于 6~8 行。如图 2-4 所示，sans serif 字体比 serif 字体具有更好的效果（serif 字体的高度较低，并带有一定的倾斜）。图 2-4 显示了两幅屏幕投影的比较（图 2-2 和图 2-3 来自于 Vetter、Ward 和 Shapiro 的著作[2]）。

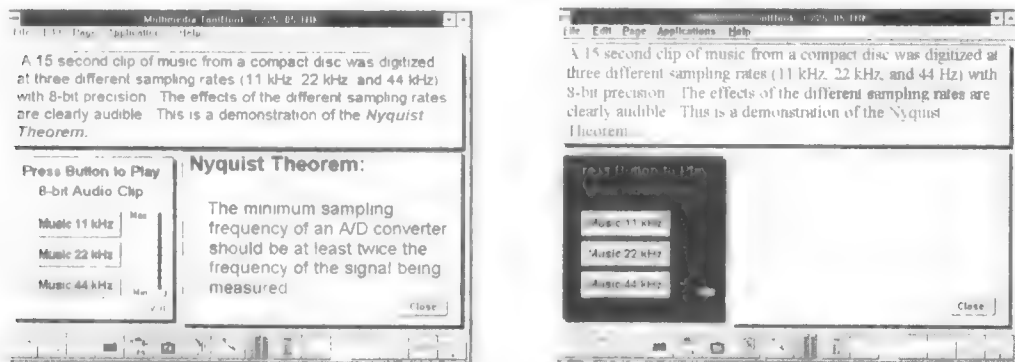


图 2-4 颜色和字体（彩色插页中也有此图），感谢 Ron Vetter

上面的图中颜色和字体使用得比较合理。它具有统一的配色方案，并使用了较大的 sans-serif（Arial）字体。底下的图则效果不佳，因为它使用了太多的颜色，而且这些颜色并不和谐。红色和蓝色在一起让人很难聚焦，因为人的视网膜很难同时聚焦于这两种颜色上。深色的 Serif(Times New Roman)字体很难辨别。此外，右下角面板的对比度不足，因为漂亮的柔和的颜色只有当其背景色有着显著不同时才能使用。

#### 4. 色彩对比程序

根据 Vetter 等人的研究成果，我们开发了一个 VB 程序来调查文本的颜色和背景颜色是如何影响文本的可读性的（这个程序可以在本书的相应网站上获得，具体信息请参看本章结尾的进一步探索一节。在网站上给出了可执行文件和源文件）。

在屏幕上构建理想的配色方案最简单的方法是使用互补色原理来找到文字的背景颜色。对于在 0~1 间（或者 0~255）取值的颜色来说，如果文本的颜色是三元组（R, G, B），那么比较合理的背景颜色可以由最大值减去这个颜色得到：



$$(R, G, B) \Rightarrow (1-R, 1-G, 1-B) \tag{2.1}$$

这里，不仅颜色在某种程度上是“相反的”（当然这和艺术家们所指的“相反”是不同的），而且如果背景的颜色较深，那么文本的颜色较浅，反之亦然。

25

在这个 VB 程序中，用户可以通过滑轨来控制背景颜色的变化，同时文本的颜色也将根据互补色原理作相应的改变。用户也可以通过单击背景打开色彩选择器来选择色彩。

如果你认为可以有更好的色彩搭配，那么单击文字打开和背景色无关的色彩选择器来挑选颜色。（文字本身也可以被编辑。）略加试验你就可以发现，某些颜色的搭配确实有更好的效果——例如粉红色背景和森林绿的前景，或是绿色背景和淡紫色前景。图 2-5 是这个程序运行时的截图。

26

图 2-6 是一个调色板（color wheel），这里相对颜色的定义为  $(1-R, 1-G, 1-B)$ 。艺术家的调色板和这个调色板有所不同，因为艺术家的调色板更多地源于艺术感觉，而非某种算法。在传统的艺术家的调色板中，黄色和品红色相对，而非图 2-6 中的蓝色；蓝色是和橙色相对的。

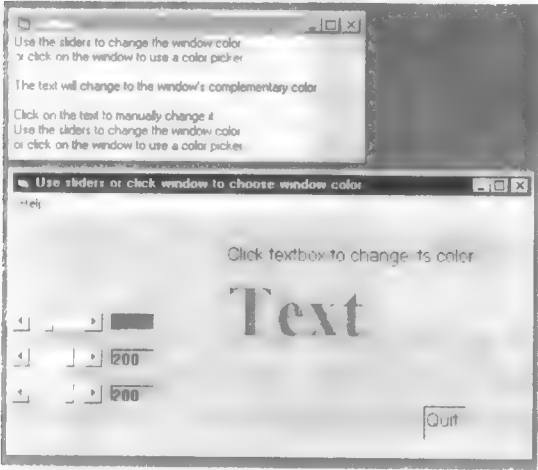


图 2-5 考察颜色和文本可读性的程序



图 2-6 调色板（彩色插页中也有此图）

5. 分镜动画

分镜经常在动画中使用。例如，在 Macromedia 的 Director 中，分镜的概念是指任何资源的实例化。然而，分镜动画的基本思想是非常简单的。假设我们已经创造了一个动画图，如图 2-7a 所示，那么可以很容易地得到一位（黑-白）的掩模  $M$ ，如图 2-7b 所示，并得到图 2-7c 所示的分镜  $S$ 。

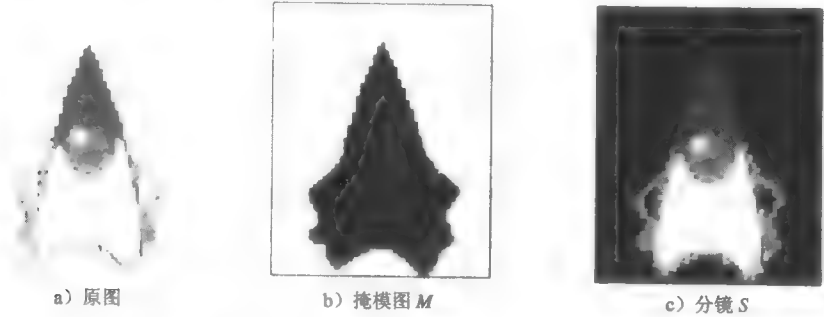


图 2-7 分镜的生成（“Duke”图来自 Sun Microsystem，在此感谢！）

现在我们可以将这个分镜覆盖在彩色的背景  $B$  上, 如图 2-8a 所示。我们需要做的是先将  $B$  和  $M$  作“与”运算, 然后将得到的结果和  $S$  作“或”运算, 最终的结果如图 2-8e 所示。将这些简单操作的组合以一定的速率执行, 就可以生成一个简单的二维动画。这个动画可以移动分镜, 但并不改变它的外观。

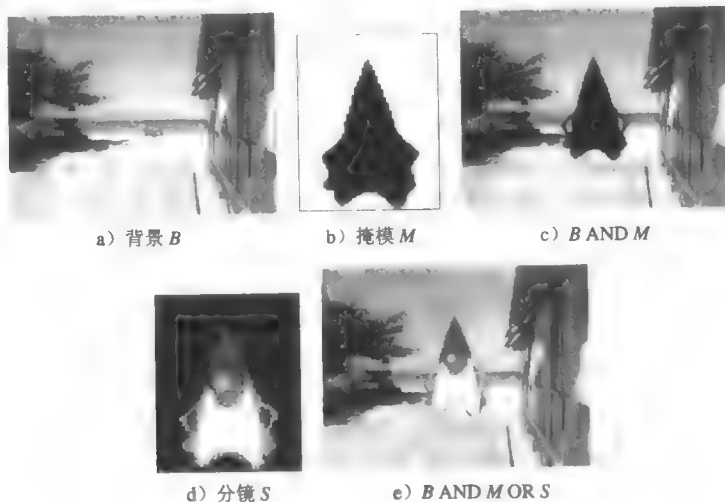


图 2-8 分镜动画

## 6. 视频的切换

视频的切换可以有效地表示切换到下一部分内容。视频切换可以作为指示“场景切换”的语法方法, 并经常带有语义信息。常用的切换类型包括: 剪切、擦拭、融化、淡入和淡出。

剪切, 顾名思义, 是在两个连续的视频帧之间执行对图像内容的剧烈变化。这是最为简单也最为常用的切换方式。

擦拭是用另一段视频中的内容来代替可视区域的像素。如果两个视频的边界在屏幕中缓慢移动, 那么第二段视频将逐渐替代第一段视频。擦拭方式有从左到右、从右到左、垂直、水平, 类似于膜孔方式, 或是类似于钟表指针掠过的方式, 等等。

融化的方式用两段视频的混合来代替原图中的像素, 以实现两段视频之间的渐变。淡出使用黑(白)色来代替视频, 淡入则相反。大多数的融化可以对应于 Adobe Premiere 视频编辑软件中的交叉融化和抖动融化归为两类。

在第一类(交叉融化)中, 像素点是渐变的。它可以被定义为

$$D = (1 - \alpha(t)) \cdot A + \alpha(t) \cdot B \quad (2.2)$$

$A$  和  $B$  是表示视频  $A$  和  $B$  的三元颜色向量。 $\alpha(t)$  是颜色切换的函数, 通常它和时间  $t$  具有线性关系:

$$\alpha(t) = kt, \quad kt_{\max} \equiv 1 \quad (2.3)$$

第二类(抖动融化)则完全不同。根据  $\alpha(t)$ , 视频  $A$  中的像素点将突然被视频  $B$  所替代, 这种变化是非连续的。变化的像素点的位置可以是随机的, 也可以遵循一定的模式。

很明显, 淡入淡出是第一类融化方式(视频  $A$  或  $B$  为黑色(或白色))的特例, 而擦拭则是第二类融化方式(采用某种几何模式来改变像素)的特例。

尽管很多数字视频编辑器都有一组预制的切换方式, 我们有时还是希望能够按照自己的意图

定制。假设我们希望创建一种特殊的擦拭模式使用滑入滑出的方式实现视频的替换。通常的擦拭方式没有这种效果。通常情况下，每段视频处于一定的位置，切换线沿着“静态”的视频运动，视区左边的部分显示左边视频的像素点，右边部分显示右边视频的像素点（对于从左到右的水平擦拭而言）。

假设我们不希望视频固定在某个位置，而是渐渐地移入（移出）视区：我们希望  $\text{Video}_L$  从左边滑入，并将  $\text{Video}_R$  推出。图 2-9 显示了这个过程。 $\text{Video}_L$  和  $\text{Video}_R$  有各自不同的 RGB 值。注意， $R$  是帧中位置  $(x, y)$  以及时间  $t$  的函数。由于这是视频而非不同大小的图像集合，因此两段视频有相同的最大范围  $x_{\max}$ 。（Premiere 将所有视频都规范为相同大小，即当前选中的区域，所以无需担心大小不同。）

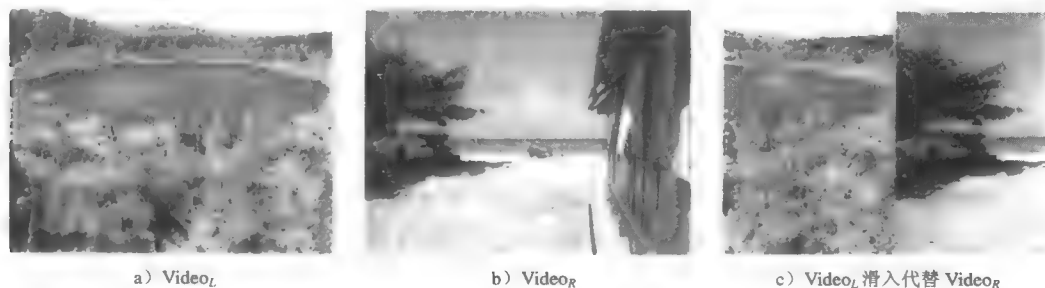


图 2-9  $\text{Video}_L$  滑入， $\text{Video}_R$  推出的过程

切换边界的水平位置  $x_T$  沿着视区从  $t=0$  时刻的  $x_T=0$  向  $t=t_{\max}$  时刻的  $x_T=x_{\max}$  运动。所以，对于线形的切换来说， $x_T = (t/t_{\max}) x_{\max}$ 。

所以对于任意时刻  $t$ ，情况如图 2-10a 所示。视区有它自己的坐标系， $x$  轴的范围是从 0 到  $x_{\max}$ 。我们将在视区内填入像素。对于每一个  $x$ （和  $y$ ），我们必须确定从哪段视频中获得 RGB 值以及从什么位置获得像素值——也就是左边视频坐标系中的  $x$  位置。根据视频的特点，左边视频的图像是不断变化的。

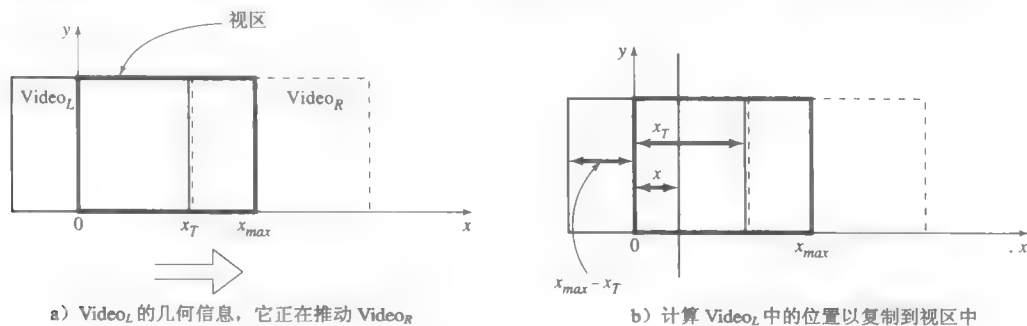


图 2-10 图 2-9 所示过程的分析

假设对  $y$  的依赖性为隐式的。在任何事件中，我们使用和源视频相同的  $y$  值。那么对于红色信道（对于蓝色和绿色的信道也相似）， $R = R(x, t)$ 。假设我们确定像素值来自于  $\text{Video}_L$ ，那么非运动视频中的  $x$  坐标  $x_L = x + (x_{\max} - x_T)$ ，其中  $x$  是我们要在视区中填充的位置， $x_T$  是切换边界的位置， $x_{\max}$  是任意一帧中的最大像素位置。

为了说明这一点，我们可以从图 2-10b 中看到，可以使用距离  $x$  与  $x_{\max} - x_T$  的和来计算视频  $L$  坐标系中的位置  $x_L$ 。

将  $x_T = (t/t_{max}) x_{max}$  带入, 我们可以得到如图 2-11 所示的伪代码。在图 2-11 中, 如果像素点来自  $Video_R$  而非  $Video_L$ , 那么也是很容易得到公式中相应的变化。

### 7. 一些技术设计问题

影响多媒体应用的设计和传输的技术参数包括计算机平台、视频格式和分辨率、存储和磁盘空间以及传输方法等。

```

for t in 0..tmax
  for x in 0..xmax
    if (  $\frac{x}{x_{max}} < \frac{t}{t_{max}}$  )
      R = RL ( x + xmax * [1 -  $\frac{t}{t_{max}}$ ], t )
    else
      R = RR ( x - xmax *  $\frac{t}{t_{max}}$ , t )

```

图 2-11 滑动视频切换的伪代码

- **计算机平台** 通常, 我们使用的计算机可以

是基于 UNIX (例如 Sun) 的计算机、PC 或者是 Macintosh。尽管不少软件表面上是“可移植”的, 但很多跨平台软件是依赖于运行时模块的, 这些运行时模块在不同系统上运行时可能会有问题。

- **视频格式和分辨率** 最为流行的视频格式为 NTSC、PAL 和 SECAM。它们互相之间并不兼容, 所以使用不同格式播放视频时需要进行转换。

在屏幕上显示像素点的图形卡有时也被称为“显卡”。事实上, 某些卡可以实现帧的抓取功能, 将模拟信号转化为用于视频的数字信号。这类卡又被称为“视频采集卡”。

图形卡的性能和它的价格有关。图形卡性能的一个较老的标准是 S-VGA, 它支持 1 280×1 024 的分辨率, 以及 16 位的 65 536 种颜色或 24 位的 16.7M 种颜色。目前, 支持 1 600×1 200 分辨率以及 32 位彩色的图形卡是非常常见的。

- **内存和磁盘空间需求** 硬件的飞速发展缓和了这个问题, 但总体来说多媒体软件的需求还是很高的。目前对于多媒体程序而言, 至少需要 128MB 的 RAM 和 20G 的硬盘空间。
- **传输方式** 在编码及其他工作都已完成后, 我们需要考虑如何显示我们的成果。由于我们已经有了足够大的磁盘, 其性能良好, 存储空间也不是什么问题, 我们可以直接带着计算机进行展现。然而, 我们通常希望将成果作为产品进行销售。目前, 可重写的 DVD 驱动并不普及, 而 CD-ROM 可能缺乏足够的容量来存储多媒体的展现文件。此外, CD-ROM 光驱的访问速度也不如硬盘。

电子传输手段是另一种方式, 这取决于用户端 (和服务端) 的网络带宽。根据展现类型, 我们可以选择是否使用流媒体技术。

对于大型多媒体项目, 目前还没有非常适用的传输机制。尽管如此, 使用 PowerPoint 或者 Director 这样的工具可以制作适合 CD-ROM 存储的多媒体展现文件。

### 2.1.4 自动编著

目前, 我们只考虑了如何编著新的多媒体。但事实上存在着很多已有的多媒体文档, 因此研究人员对自动编著 (automatic authoring) 的方法一直有着很浓厚的兴趣。这个术语可以用于表示开发多媒体展现的高级帮助工具, 也可以表示利用已有的资源开发多媒体文档的自动创建机制。

#### 1. 超媒体文档

首先我们考虑超媒体文档。通常, 产生文档需要三个步骤: 信息的生成和捕捉、编著以及发布 (这三个步骤可以看成是非线性的)。但问题是, 在这个过程中多少内容可以自动完成?

第一个步骤是媒体的捕捉, 如果是来自文本或是音频数字器或是视频帧抓取器, 那么就可以很方便地实现自动化。最后一步 (多媒体的展现) 则是我们考虑的多媒体工具的目标。中间的步骤 (编著) 则是我们考虑的重点。

从本质上说, 我们希望能将信息结构化以支持对可用媒体的访问和操纵。显然, 我们可以考虑在这里使用标准的计算机数据结构——链表、树或网络 (图)。不过, 这里我们将考虑如何更好地

实现数据的结构化以支持多视图而非单一的静态视图。

## 2. 外表化和线性化

图 2-12 显示了缺乏超媒体支持的通信方式的本质问题：作者的思维被线性化了。相反，超链接则可以帮助我们部分地模仿作者的思维过程（即外表化）。毕竟，1.2.1 节中 Bush 的 Memex 思想的本质涉及人类记忆的相关链接。

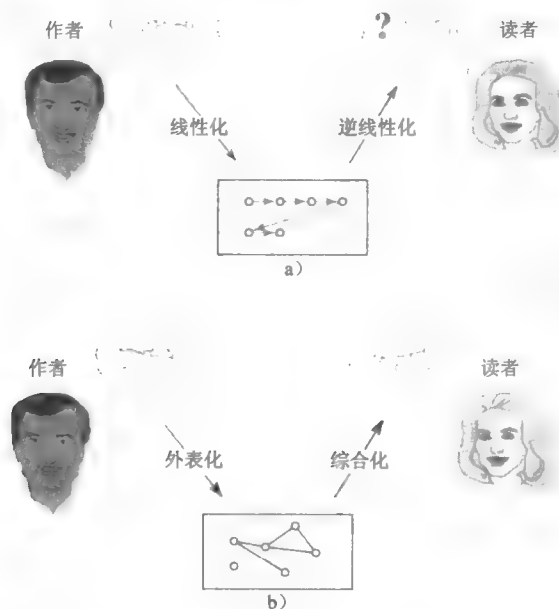


图 2-12 使用超链接的通信（感谢 David Lowe (©1995 IEEE)[5]）

以 Microsoft Word 为例，现在我们可以很方便地创建一份文档的超文本版本，因为 Word 本身就有章和标题这样的结构。但如果我们提取语义内容以及发现链接和定位点，那么就会产生问题，即使只考虑文本不考虑图像。图 2-13 显示了这个问题：尽管管理一些信息节点是可行的，但当问题的规模变得很大时，我们就需要自动助手的帮助。

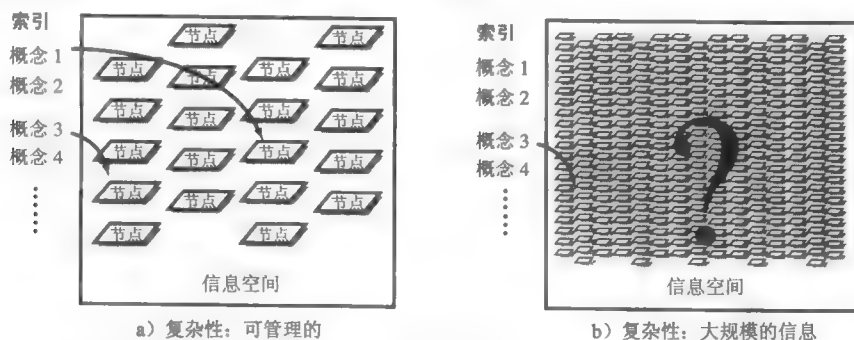


图 2-13 复杂信息空间（感谢 David Lowe (©1995 IEEE)[5]）

当数据集的规模变大时，我们就要应用数据库的方法。此时需要关注的问题有扩展性（扩展为大数据集）、可维护性、资料的添加和可重用性。数据库的信息必须按以下方法设置：“发布”阶段（向用户展现）可以采用及时方式，并能通过在中间信息结构中定义的用户视图来表达信息。

### 3. 超文本的半自动迁移

文本信息的超链接结构非常简单：“节点”表示语义信息并且是指向其他页面的定位点。图 2-14 显示这些概念。

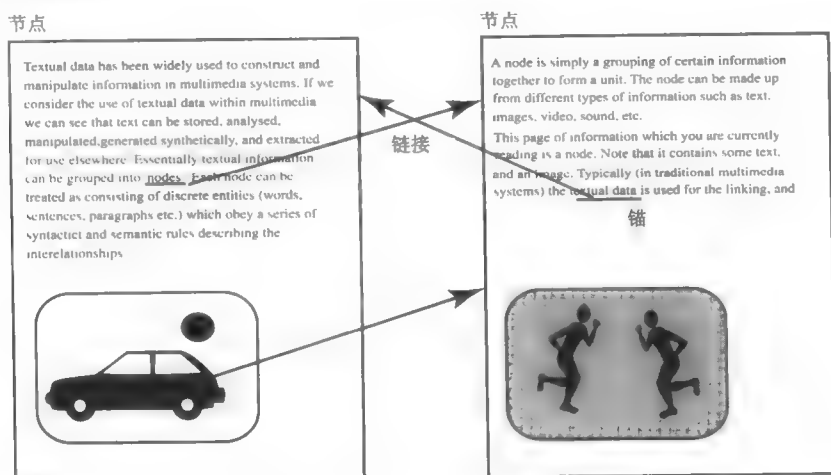


图 2-14 超文本中的节点和定位点（感谢 David Lowe [6]）

对于文本，从基于纸面的信息向超文本进行迁移的第一个步骤是将格式自动转化为 HTML。然后，章节信息可以保存在数据库中。然后，可以应用简单的数据挖掘技术（按词干检索）很方便地对标题和题目进行分析以提取关键词——如通过频率统计。关键词可以被加入到数据库中。然后，辅助程序就可以在相关概念之间自动生成附加的超链接。

这类程序的一个半自动版本最有可能获得成功，它提供被接受或拒绝，并且能够手动添加的建议。当插入新的节点时，数据库管理系统可以维护链接的完整性。对于发布阶段，考虑到重建底层信息结构的难度，因此我们希望能尽可能晚地在数据上施加视点（viewpoint）。

### 4. 超图像

考虑图像或是其他多媒体数据时，问题就不是那么直观了。如果希望使用类似于对象文本的方法来对待图像，我们就要将图像看成是包含对象和其他锚的节点，并确定图像的实体和规则。我们需要的是能够帮助我们生成真正的超媒体的自动方法，如图 2-15 所示。

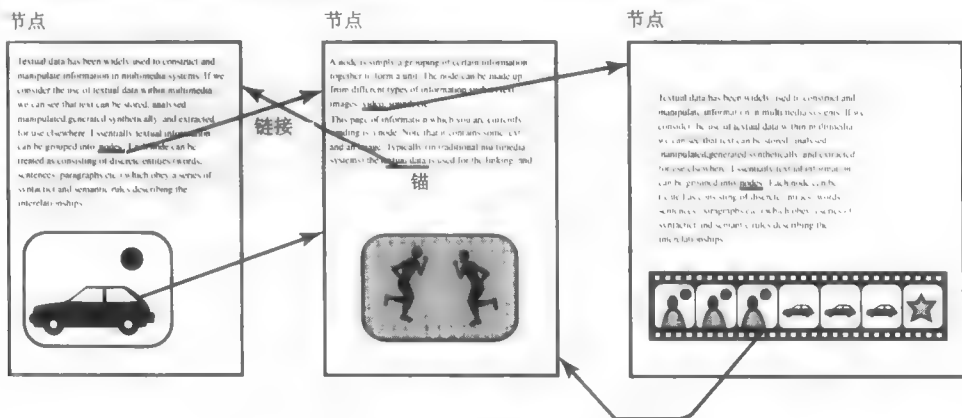


图 2-15 超媒体结构（感谢 David Lowe[6]）

通过屏蔽图像区域，我们可以手动地描述图像的语法元素，它们可以用文本进行标记，这样我们就可以使用前面基于文本的方法。图 2-16 显示了“超图像”，它具有一个确定的图像区域，可以自动地链接到文档的其他部分。

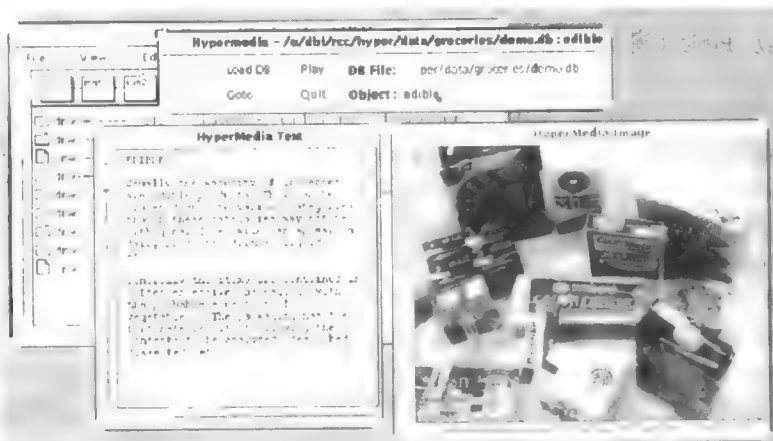


图 2-16 超图像（感谢 David Lowe [6]）

尽管这些方法还处于起步阶段，但已经展现出了自动编著的美好前景。自然的，我们还希望了解哪些数据库系统、数据挖掘、人工智能等方面的工具可以用来帮助我们开发功能全面的多媒体系统，而不仅仅是超媒体系统。上面的讨论说明我们确实还只是处在起步阶段。

35  
~  
36

## 2.2 多媒体编辑和编著工具

本书重点关注多媒体的原理，即真正了解这一主题所需的基础内容。然而，我们需要某种载体来体现对这一内容的理解，而 C++ 程序或是 Java 都不是最佳的选择。大多数多媒体导论课程都要求你至少能够完成一些多媒体作品（参见练习 11），所以我们需要一个“助推器”来帮助你学习“其他的软件工具”，本节就将介绍这个“助推器”。

因此，我们将考虑一些流行的编著工具。因为创建多媒体应用的第一个步骤是创建有趣的视频片段，所以我们将首先考查视频编辑工具。尽管这并不是真正的编著工具，但由于视频创作的重要性，我们将对这类程序做一下简单介绍。

我们将要介绍的工具包括：

- Adobe Premiere 6
- Macromedia Director 8 和 Macromedia Director MX
- Flash 5 和 Flash MX
- Dreamweaver MX

这些工具常常用于开发多媒体内容，当然这里并没有列出所有可用的工具。

### 2.2.1 Adobe Premiere

#### 1. Premiere 的基础知识

Adobe Premiere 是一款非常简单的视频编辑程序，用户可以通过组合和合并多媒体组件来快速地创建简单的数字视频。它有效地使用了乐谱创作模式，在乐谱创作模式下，所有的组件都被水平排放在时间轴窗口中。



通过“File→New Project”命令可以打开一个窗口，窗口中包含一系列的预置信息——包括帧的分辨率、压缩方法和帧率的值的集合。此外还有很多预置的选项，大多遵循 NTSC 或 PAL 视频标准。

首先可以导入资源，如 AVI (Audio Video Interleave, 音频视频交叉) 视频文件和 WAV 声音文件，并将它们从 Project 窗口中拖拉到轨道 1 或 2 上。(事实上，你最多可使用 99 条视频轨道和 99 条音频轨道!)

视频 1 实际上包含三个轨道：Video 1A (视频 1A)，Video 1B (视频 1B) 和 Transitions (切换)。Transitions 可以仅仅应用于 Video 1。Transition 可以由 Transition 窗口拖拉到 Transition 轨道上，例如用 Video 1B 渐渐替换 Video 1A (融化)，棋盘格中随机像素点的突然替换 (抖动融化)，或是擦拭 (一段视频滑入替代另一段视频)。程序中有许多可选的切换方式，你也可以使用 Premiere 的 Transition Factory 来设计独创的切换方式。

37

可以通过将 WAV 声音文件拖拉到时间轴窗口中的 Audio1、Audio2 或者任何其他的声音轨道来导入 WAV 声音文件。你可以通过右键单击来编辑声音轨道的属性。

图 2-17 显示了一个典型的 Adobe Premiere 界面。时间轴窗口顶部的黄色标尺表述了当前工作的时间轴——可以通过拖拉该标尺到达合适的时间位置。底部的 1 Second 下拉框表示目前的视频帧率为每秒一帧。



图 2-17 Adobe Premiere 的界面

要对视频进行“编译”，使用“Timeline→Render Work Area”并将工程保存为.ppj 文件。现在你需要作一些选择，包括如何以及用何种格式来保存视频。图 2-18 显示了工程的选项。包含编解码方式的对话框是由编解码生产商提供的，我们可以单击 Configure 按钮来得到它们。压缩编解码 (压缩-解压缩协议) 通常位于视频采集卡的硬件上。如果你选择了需要硬件支持的编解码算法，那么别人的系统就可能无法播放你的漂亮的数字视频，一切工作都是徒劳。

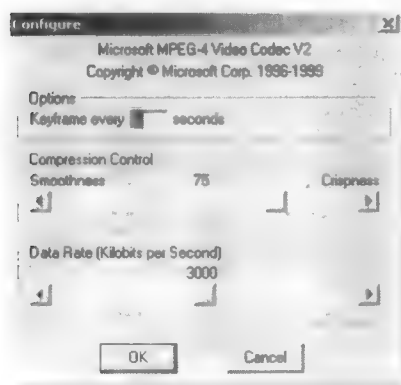
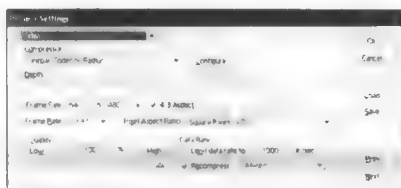
图像也可以被插入到轨道中。我们可以使用切换方式让图像在最终的视频窗口中渐渐出现或是渐渐消失。为了实现这一目的，我们需要建立一个“掩模”图像，如图 2-19 所示。这里，我们导入了 Adobe Photoshop 的层次图像，它具有在 Photoshop 中创建的 alpha 通道。

接着在 Premiere 中，我们单击位于视频轨道中的图像，并使用“Clip→Video Options→Transparency”来设置 Alpha 通道的键 (它将触发透明性)。也可以使用 Clip→Video Options→Motion 来实现图像在视频帧中飞入飞出。

38

在 Photoshop 中, 我们通过下列步骤设置 alpha 通道:

- 1) 使用你喜欢的图片, 如 .JPG 文件。
- 2) 用单色作为背景, 如白色。
- 3) 选择 “Image→Mode→RGB Color”。
- 4) 选择背景区域 (你希望在 Premiere 中保持不透色的区域), 可以使用魔术棒工具来完成。

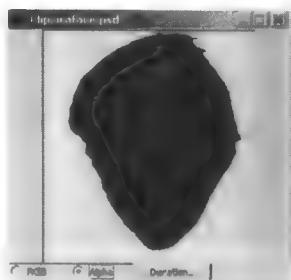


b) 压缩选项

图 2-18 设置选项



a) RGB 通道



b) alpha 通道

图 2-19 掩模

- 5) 选择 “Select→Save Selection...”。
- 6) 确定 “Channel = New”。按下 “OK”。
- 7) 选择 “Window→Show Channel”。双击新的通道, 并将其重命名为 Alpha。将其颜色设置为 (0, 0, 0)。
- 8) 将其保存为 PSD 文件。

如果在 Photoshop 中创建的 alpha 通道具有白色的背景色, 那么当你在 Premiere 中选择 Alpha 时需要选择 Reverse Key。

Premiere 有比较简单方法来为数字视频创建标题 (在需要荣誉的地方给予荣誉)。

Premiere 另一个很不错的特点是它可以很方便地捕捉视频。为了从录像带或摄像放像机中得到一段数字视频, 选择 “File→Capture→Movie Capture” (视频/音频捕捉选项的菜单可以通过右键单击捕捉窗口得到)。类似的, 将其存为模拟格式也非常简单。

## 2. Premiere 的切换

Premiere 提供了多种有趣的视频切换方式。然而, 通过对结果视频进行逐帧的检查会发现内建的切换方式并不像 “广告” 所说的那样工作。例如, 通过仔细的检查, 我们发现一种声称是基于时间的线性擦拭方法在一开始就是非线性的——视频切换线的运动并不是匀速的。

如果感兴趣的话, 我们可以使用 Premiere 的 Transition Factory 为我们提供的大量的函数来创建自己的切换方式。由于我们实际上处于一种 int 的状态下, 因此这些函数 (如 sin 和 cos) 都具

有整数（而非浮点数）的值域和定义域。所以在使用时我们需要注意这个问题。习题 9 给出了实际问题中的一些详细信息。

## 2.2.2 Macromedia Director

### 1. Director 的窗口

Director 是一个用来创建互动式“电影”的完整环境（见图 2-20）。它使用了电影模式，程序中的窗口就显示了这一点。动作发生的主窗口称为舞台（Stage）。显式地打开舞台将会自动关闭其他所有窗口（一种有用的快捷方式是 Shift+Keypad-Enter（数字键盘边上的回车键，并非通常意义上的回车键）；这将清除舞台窗口之外的所有窗口，并开始电影的播放。）

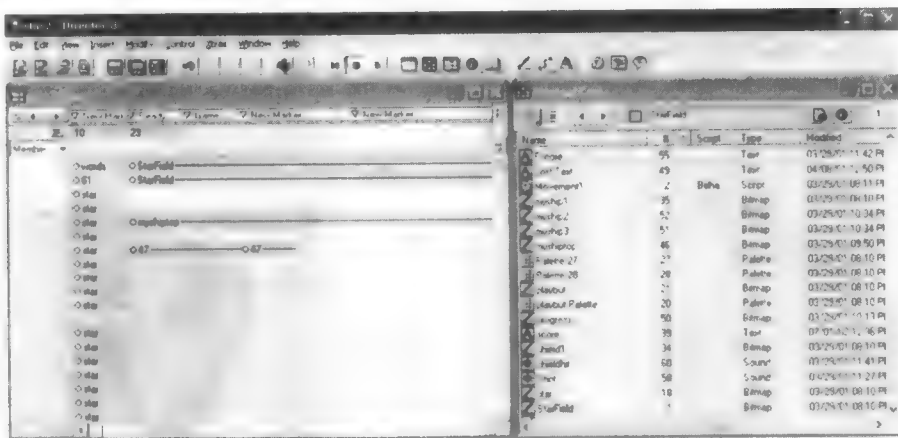


图 2-20 Director 的主窗口

另外两个主要的窗口是 Cast 和 Score。Cast 中包含影片会用到的资源，如位图、声音、向量图、Flash 电影、数字视频和脚本。Cast 中的成员可以直接创建或从其他位置导入。通常，你可以创建多个 Cast，以便更好地组织电影的不同部分。可以从 Cast 窗口中拖拽 Cast 成员，将其放在舞台上。因为一个 Cast 成员要用到多个实例，所以每个实例称为一个分镜。一般而言，Cast 成员是原始媒体，而分镜则是控制在舞台和影片上 Cast 成员在何时、何地以及如何出现的对象。

向分镜中添加预定义或专门的“行为”（例如让分镜跟踪鼠标）可以使其具有交互性。行为位于 Director 的内部脚本语言中，这种语言称为 Lingo。Director 是一种标准的事件驱动程序，可以方便地进行对象定位并向对象添加事件过程。

有许多预定义的事件集，其中包含鼠标事件以及网络事件等（后者的一个例子是测试角色成员是否被成功下载）。可行的控制方式是在部分展现中不断循环直到视频下载完成为止，然后继续或跳转到另一帧。位图可以用来作为按钮，最常见的用途是在单击按钮后跳转至另外一帧。

乐谱窗口由一组水平线（一条线对应一个分镜）和垂直的帧组成。因此，乐谱窗口看起来和音乐的乐谱比较相似，也是按时间顺序从左至右排列，但是它更像 MIDI 文件中的事件链表（参见第 6 章）。

预定义和用户定义的行为类型均在 Lingo 中。调色板库提供了对所有预定义行为脚本的访问方式。你可以向一个分镜中添加行为或向整个帧中添加行为。

如果行为中包含参数，那么就会出现一个对话框。例如，对于网页浏览行为，我们需要指定要跳转的帧。你可以将同样的行为附加给许多分镜或帧，并为每个实例指定不同的参数。大多数行为都能响应简单的事件，如在分镜上的单击或是在“放音磁头”进入一帧时触发的事件。大多

数的基本功能（如声音的播放）都被封装在程序包中。开发自定义的 Lingo 脚本则能提供更多的灵活性。我们可以在 Inspector 窗口（包括 Behavior Inspector 和 Property Inspector）中修改行为。

## 2. 动画

传统的动画是按时间先后显示稍有不同的图像实现的。在 Director 中，这一方法意味着在不同的帧中使用不同的成员。为了便于控制这个过程，Director 允许将多个成员合并为一个单独的分镜（如果要在乐谱上显示，选择所有需要合并的图像，使用 Cast To Time 菜单项使它们显示在当前的乐谱位置上）。一个有用的特点是，通过扩展这类动画乐谱上的时间可以减缓每一幅图像的播放时间，这样整个动画就可以按照规定的时间长度来完成播放。

另一种相对简单的动画方法是使用 Director 的渐变（tweening）功能。这里，你需要在舞台上移动某幅图像并保持原图不变。“渐变”是初级动画人员的工作，他们主要负责在高级动画人员创作的关键帧之间进行内容填充——在 Director 中是自动完成的。

为了实现这种动画，在舞台上确定渐变帧的路径。你也可以确定一些关键帧以及关键帧之间的曲线变化。你还需要确定在运动的开头和结尾处图像如何加速和减速（“缓入”和“缓出”）。图 2-21 显示了渐变的分镜。

另一类广泛使用的简单动画形式称为调色板动画。如果图像是 8 位的，那么在颜色查找表中循环搜索或是系统地对查找表的条目进行替换可以得到非常有趣（或是奇异）的效果。

乐谱窗口的重要特性包括通道、帧和播放头。后者表示我们在乐谱中的位置。单击乐谱中的任意位置将对播放头重新定位。通道是乐谱中的行，可以包括可见媒体的分镜实例。因此，这些编号的通道又称为分镜通道。

在乐谱窗口的顶端是用来控制调色板、速度、切换和声音的特效通道。图 2-22 显示了 Score 窗口中的这些通道。在分镜和特效通道中，帧被水平编号。和传统电影一样，帧是影片中的一个独立单元。我们可以通过修改每秒的帧数来控制影片的播放速率。

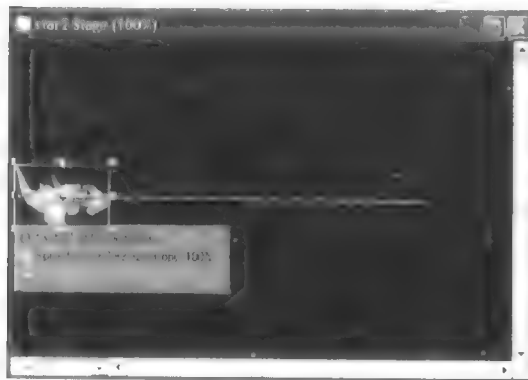


图 2-21 渐变的分镜

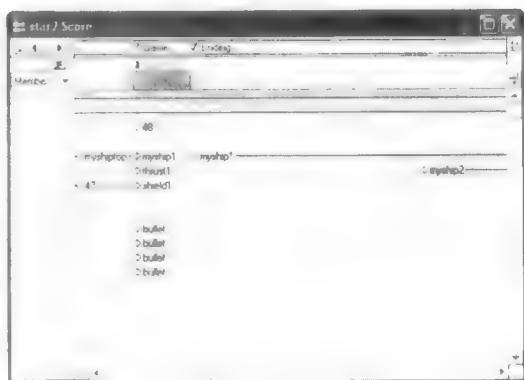


图 2-22 Score 窗口

## 3. 控制

你可以在任意帧中加入命名标记。最简单的控制事件就是跳转至特定的标记。在 Director 中，每个标记是一个场景的开始。触发帧浏览的事件是 Go To Frame、Go To Marker 或者 Hold on Current Frame。Hold on Current Frame 将使影片停止在当前帧的位置。帧的行为将在乐谱窗口的脚本通道中显示。

按钮就是附加了行为的位图。你通常需要两幅位图，分别刻画按钮按下和放开时的状态。内建的 on mouseUp 事件则产生跳转。

#### 4. Lingo 脚本

Director 使用四种类型的脚本：行为、演员的附加脚本、电影脚本和父脚本。行为、电影脚本和父脚本在演员窗口中都作为演员出现。

“行为”是附加到分镜或帧上的 Lingo 脚本。根据用户是否单击按钮，你可以使用脚本来确定分镜是否移动。一个非常有用的特点是脚本可以根据多媒体资源的下载程度来控制其播放的开始时间。如果希望添加行为，可直接将其从一个演员拖拽到乐谱或舞台中的一个分镜或一帧上。

我们也使用电影脚本，它对于整个影片都是可用的。电影脚本可以控制当电影开始、结束或暂停时的事件响应，此外还可以对诸如按键或鼠标单击这样的事件进行响应。父脚本可以用来为一个对象创建多个实例，而无需向乐谱中添加演员。

用户定义的 Lingo 脚本可以用来创建动画或对典型事件（如用户利用键盘和鼠标产生的动作）进行响应。脚本同样可以用来流化视频、浏览网页以及格式化文本等。

Lingo 脚本还可以在乐谱的基础上对行为加以进一步的扩展。基本的数据类型是链表，这是最基本的数据结构。你同样可以使用链表进行数组的操作。此外，还有数学运算和字符串操作。链表具有两种类型：线形和属性。

线形链表就是 LISP 中的链表，例如{32, 43, 12}。属性链表是一个关联列表，这种列表也和 LISP 中的链表类似：每个元素都包含两个用冒号隔开的值。每个属性前都有一个数字符号。例如，下面的语句生成了两个规定分镜坐标的属性链表：

```
sprite1Location = [#left:100, #top:150, #right:300, #bottom:350]
sprite2Location = [#left:400, #top:550, #right:500, #bottom:750]
```

Lingo 具有很多对链表进行操作的函数。例如，append 可以在链表的结尾增加一个元素，deleteOne 可以从链表中删除一个值。

##### Lingo 规范

- 函数 the frame 表示当前帧。
- 特殊标记 next 或 previous 表示相邻的标记（不是相邻的帧）。
- 函数 marker(-1) 返回前一标记的标识符。如果当前帧被标记并有标记名，那么 marker(0) 返回当前帧的标记名；否则，它返回前一标记的名称。
- Movie "Jaws" 表示名为“Jaws”的全局电影的起始帧。通常它是另一个 Director 电影的名字。引用 frame 100 of movie “Jaws” 指向这部电影。

这些细节在在线帮助的 Lingo 部分都已详细列出。帮助目录“Learning→Lingo\_Examples”下有许多 DIR 文件详细描述了 Lingo 使用的基础知识。

#### 5. Lingo 的窗中电影

作为 Lingo 应用的精彩例子，关于如何创建窗中电影的 Lingo 帮助文档概括介绍了如何添加脚本。

Lingo 是一种标准的事件驱动程序语言。事件处理程序附加到特定的事件，如 mouseDown 消息。脚本中包括事件处理程序。你可以通过将脚本附加到对象来实现事件处理程序和对象的绑定。

#### 6. 3D 分镜

Director 新增的一项功能是在舞台中创建、导入和操纵 3D 对象。可以加入到 Director 的一个简单的 3D 对象是 3D 文本。为了创建 3D 文本，首先选中任意的普通文本，然后在属性查看器（Property Inspector）中单击 Text（文本）选项卡，将显示模式设为 3D。其他选项，如文本深度和纹理，可以通过属性查看器中的 3D Extruder 选项卡来改变。这些属性也可以在 Lingo 中动态设置，使文本可以根据电影的进度而改变。

除文本外的其他 3D 对象只有通过 Lingo 实现，或从 3D Studio Max 中导入。Director 支持很

多基本的 3D 动画元素, 包括如球形和用户定义网格这样的基本形状。可以为这些基本形状加入纹理和阴影。纹理是加在 3D 模型上的 2D 图像, 而阴影则定义了基本模型的外观。我们还可以在场景中加入光线。默认情况下, 整个场景中都充满了光线。可以增加的光线有四类: 充满的、有向的、点式的和聚焦式的。也可以指定光线的强度和颜色。

用户的视角, 又称为摄像机, 可以进行移动以便从不同角度展示 3D 对象。摄像机的运动(如水平摇动和垂直倾斜)可以通过库(Library)窗口的内建脚本来控制。

### 7. 属性和参数

为 Lingo 行为定义行为参数可以增加创建时灵活性。参数可以通过在行为创建时提供输入来改变行为。如果没有定义参数, 那么就会使用默认值。改变一个特定行为的参数是很容易的, 只需在它附加到另一个演员时双击行为的名字就可以了。图 2-23 显示了改变参数的对话框。

行为可以具有名为 `getPropertyDescriptionList` 的特殊处理程序, 当创建附加于行为的分镜时它开始运行。处理程序返回一组可以加入到 `addProp` 函数中的参数。例如, 如果 Lingo 创建一个运动行为, 那么可以加入参数来规定运动的方向和速度。行为可以附加于多个演员上为运动进行说明。

`getPropertyDescriptionList` 中定义的参数是通过该行为任何句柄都可以访问的行为的属性。我们可以通过在处理程序外使用 `property` 关键字并列所有属性(用逗号分隔)来定义行为的属性。全局变量可以通过行为进行访问。它们可以像属性那样被声明, 只是要用 `global` 关键字。任何需要访问全局变量的行为必须用 `global` 关键字加以声明。

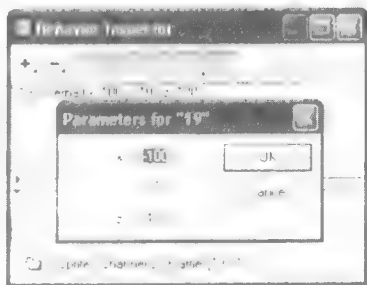


图 2-23 Parameters 对话框

45

### 8. Director 对象

Director 主要有两类对象: 一类在 Lingo 中创建, 另一类是在乐谱中创建的。父脚本可以用来在 Lingo 中创建新对象。通过在属性查看器中改变脚本的类型, 我们可以将行为转化为一段父脚本。父脚本和其他行为不同, 当在 Lingo 脚本中创建父脚本时, 参数将传递给对象。

父脚本只能在 Lingo 中创建和修改, 而乐谱中的对象只能被操纵。最常用的对象是乐谱中的分镜。分镜只能和引用它们的 Lingo 脚本在同一时段内使用。可以使用 `Sprite` 关键字加上分镜通道号来指定在某个通道对分镜进行引用。

分镜具有很多属性, 这些属性用于执行不同的行为。`locv` 和 `loch` 属性可以分别改变分镜的垂直和水平位置。`member` 属性则规定了分镜中的演员并可以改变附加于行为上的演员。这在动画中是非常有用的, 可以在 Lingo 中改变分镜来反映细微的变化, 而无需在乐谱中完成。

## 2.2.3 Macromedia Flash

Flash 是用来创作交互式电影的一个简单的编著工具。Flash 采用乐谱模式来进行电影的创作和窗口的组织。在本节中, 我们将简单介绍 Flash 并提供一些有关其使用的例子。

### 1. 窗口

电影是由一个或多个场景构成的, 每一个场景都是电影中的一个独立部分。利用“Insert→Scene”命令可以在当前的电影中创建一个新的场景。

在 Flash 中, 构成电影的组件(如图像和声音)称为符号(Symbol), 我们可以通过把符号放置到舞台上将其添加到电影中。舞台是位于屏幕中央窗口中的一个始终可见的大的白色矩形。Flash 中另外三个重要的窗口包括时间轴(Timeline)、库(Library)和工具(Tools)。

46

## 2. 库窗口

库窗口中显示当前场景中的所有符号，并可以用“Window→Library”命令来切换这些符号。可以通过双击库窗口中符号的名字来编辑，使其出现在舞台中。如果要添加符号，只需将其从库中拖拽到舞台中。

## 3. 时间轴窗口

时间轴窗口控制场景的层次和时间轴。时间轴窗口的左边部分由舞台中的一或多层组成，可以使你方便地对舞台内容进行组织。库中的符号可以被拖拽到舞台中的某个层次上。例如，一个简单的电影可以有两个层次，即背景和前景。当选中背景层时，库中的背景图形可以被拖拽到舞台中。

对于层而言，另一个有用的功能是锁定或隐藏层。点击层名称旁边的圆形按钮，可以在层的隐藏/锁定状态间切换。当在另一个层中定位或编辑一个符号时，隐藏层非常有用。层完成之后，可以锁定该层以免其中的符号被意外修改。

时间轴窗口右侧由场景中每一层使用的水平栏组成，和乐谱很相似。它表示影片经过的时间。在不同的层中，时间轴包含一些关键帧。按 F6 键可以在当前层中插入关键帧。启动一个动画或一个新符号的出现这样的事件必须在关键帧中设置。点击时间轴会改变当前编辑的影片的时间。

## 4. 工具窗口

工具窗口主要用来进行图像的创建和操作，它主要由四部分构成：工具(Tools)、视图(Views)、色彩(Colors)和选项(Options)。工具中包括选择工具，可以用来对现有的图片进行分割，此外还有一组简单的绘图工具，如铅笔和油漆桶等。视图中包括缩放工具和指针工具，可以用这些工具在舞台中浏览。色彩工具则用于选取前景和背景色，并标记出需要操作的颜色。选项工具则用于在选中某种工具时提供附加选项。

有许多其他的窗口也可用于操纵符号。除时间轴窗口用 View-Timeline 命令切换外，其他窗口都可以在 Window 菜单下切换。图 2-24 显示了基本的 Flash 界面。



图 2-24 Macromedia Flash



## 5. 符号

符号可以由其他符号组成，或是通过绘制和导入得到。Flash 可以向符号库中导入多种音频、图像或视频格式。可以使用“File→Import”命令来导入符号，并将其自动添加到当前库中。按下 Ctrl+F8 可以为影片添加一个符号。此时，会出现一个弹出式对话框，你可以在这个对话框中指定符号的名称和行为。符号可以有以下三种行为：按钮、图形或电影。符号（例如一个按钮）可以使用工具窗口来绘制。

47

## 6. 按钮

为了创建一个简单的按钮，可以利用按钮行为创建一个新符号。时间轴窗口应该具有四个关键帧：up、down、over 和 hit。这些关键帧显示当特定动作发生时按钮的不同外观。只有 up 关键帧是必须的，它也是默认的，其他关键帧都是可选的。可以按如下方法绘制按钮：选择工具窗口中的矩形工具，然后将一个矩形拖拉到舞台中。

为了使按钮的外观能够在事件触发时发生改变，我们可以通过单击相应的关键帧来创建按钮图像。在至少定义一个关键帧之后，我们就完成了一个基本的按钮，尽管此时尚未添加任何动作。动作将在后面的“动作脚本”一节中加以讨论。

利用其他符号来创建符号和创建场景类似，也就是将所需的符号从库中拖拉到舞台中。这将实现通过简单符号来创建复杂符号。图 2-25 显示了创建符号的对话框。

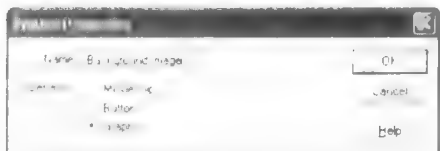


图 2-25 创建标记对话框

## 7. Flash 动画

可以通过在符号的关键帧之间生成细微的差别来获得动画。在第一个关键帧中，我们可以把需要产生动画的符号从库中拖拉到舞台上。然后插入另一个关键帧，使符号发生变化。这个过程可以不断重复。尽管这个过程比较耗时，但比其他动画技术具有更大的灵活性。Flash 同样支持以其他更为简单的方式创建特定的动画。渐变可以生成简单的动画，这种方式可以自动完成关键帧之间的变化处理。

48

## 8. 渐变

渐变有两种形式：形状渐变和运动渐变。形状渐变可以用来创建一个形状，该形状会随着时间改变为其他形状。运动渐变则允许你在舞台中的不同关键帧中的不同位置放置一个符号。Flash 自动在起始点和结束点之间填充关键帧。要执行运动渐变，首先选择要进行渐变的符号，然后选择“Insert→Create Motion Tween”命令，以及结束帧。接着使用“Insert→Frame”命令，并将符号移动至需要的位置。更高级的渐变可以对路径和加速进行控制。运动渐变和形状渐变结合在一起可以获得其他的效果。

掩模动画涉及对掩模层的操作，掩模层是一个选择性地覆盖其他层的一部分的图层。例如，为了获得爆炸效果，可以使用掩模来覆盖除爆炸中心外的其他所有区域。形状渐变可以扩大掩模，这样就可以看到整个爆炸效果了。图 2-26 显示了加入渐变效果前后的场景。

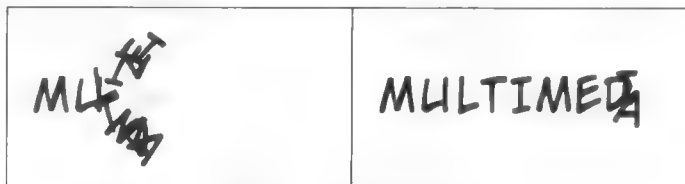



图 2-26 字母渐变前后

## 9. 动作脚本

49 上。此时将会出现 **Frame Action**（帧的动作）窗口，左应用的动作标记。动作脚本可以分为六类：**基本动作、动作、运算符、函数、属性和对象**。图 2-27 显示了帧动作窗口。

基本动作 (Basic Actions) 可以用于为影片添加一些简单的动作。常用的动作包括:

- **GOTO** 使影片前进到规定的关键帧处，并可以停止。停止动作通常用来让用户停止交互式电影的播放。
  - **Play** 如果影片停止，继续播放影片。
  - **Stop** 如果影片正在播放，则停止影片的播放。
  - **Tell Target** 向 Flash 中的不同符号和关键帧发送消息。它通常用来开始或停止不同符号或关键帧上的动作。
- 
- 图 2-27 动作脚本窗口

动作 (Action) 类型中包含很多程序结构, 如 Loops 和 Goto 语句。此外还包括其他动作, 和常用的高级事件驱动的编程语言 (如 Visual Basic) 非常类似。运算符 (operator) 类型则包括变量的比较和赋值运算符, 使你可以在动作脚本中对变量进行运算。

50 (Object) 类型列出所有的对象, 如影片片段或字符串以及它们的相关函数。

按钮需要动作脚本（事件过程），这样按下按钮才会有某种效果。将一个动作（如重放 Flash 影片）直接附加给按钮是很简单的。选中按钮并单击已打开位于屏幕右下角的动作脚本窗口。然后单击 Basic Actions，将产生一个下拉式动作列表。双击 Play 动作会自动将它添加到窗口的右边。现在单击这个按钮将会实现影片的重放。

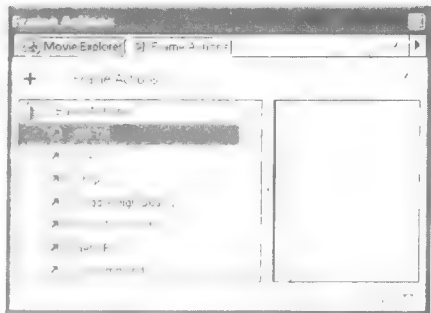


图 2-27 动作脚本窗口

### 2.2.4 Dreamweaver

Dreamweaver 是一款非常流行的 Macromedia 产品（当前版本是 Dreamweaver MX），可以用来开发支持多媒体的网页和基于 HTML、XML 以及其他格式的 Internet 应用程序。它提供了可视化的布局工具以及编辑 JavaScript、Active Server Pages、PHP 和 XML 这些文件格式的代码的能力。该产品和其他的 Macromedia 产品（如 Flash MX 以及 Fireworks MX）集成在一起。

除了作为基本的 WYSIWYG 网页开发工具之外，Dreamweaver 另一个和编著有着更为直接联系的部分是它具有一组预封装的行为，并且这些行为是可扩展的。这些行为从本质上来讲是事件过程，可以对鼠标移动这样的事件进行响应——可能的事件集对于不同的目标浏览器来说是不同的，并且可以根据版本号和浏览器重新配置。计算机科学专业的学生可以编写自己的 JavaScript 代码并将其附加到事件上。

## 2.3 VRML

### 2.3.1 概述

VRML (Virtual Reality Modeling Language, 虚拟现实建模语言) 诞生于第一届国际万维网会

议。Mark Pesce、Tony Parisi 和 David Ragget 在会议上提出了 VRML 的结构, 并且确定它将是应用在 Internet 上的与平台无关的语言。VRML 的目标是将有色彩的对象置于 3D 环境中。

VRML 是一种解释性语言, 这可以看做是它的一个劣势, 因为它在很多计算机上的运行速度比较慢。然而, 它确实很有影响力, 因为它是第一种可以用来在万维网上展现 3D 世界的方法。

严格地说, VRML 并不是像 Premiere 或 Director 那样的工具。事实上, 创建 VRML 内容唯一需要的软件就是一个文本编辑器。不管如何, VRML 是用来在 Web 环境中创建 3D 环境的工具, 就像 Flash 是用来创建交互式影片的工具一样。

### 1. 历史

VRML 1.0 在 1995 年 5 月发布, 随后在 1996 年 1 月又发布了修订版本 VRML 1.0C。VRML 是基于 Silicon Graphics 公司开发的文件发明者格式的一个子集。VRML 1.0 支持多种简单 3D 对象的创建, 如立方体、球体和用户定义的多边形等。还可以为对象指定材料和纹理使之更为逼真。

VRML 最近的一个重要的修订版本是 VRML 2.0, 这个版本增加了创建交互世界的能力。VRML 2.0 也称为“运动的世界”, 支持在交互式虚拟世界中的动画和声音。我们可以添加新的对象以便更加容易地创建虚拟世界。VRML 中引入了 Java 和 Javascript, 以支持交互式对象和用户定义的动作。VRML 2.0 相对 1.0 版本有了很多的改变, 它们之间是互不兼容的。然而, 我们可以通过某些方法将 VRML 1.0 转化为 VRML 2.0。

VRML 2.0 作为标准被提交给国际标准化组织 (ISO), 结果产生了 VRML 97。实际上, VRML 97 和 VRML 2.0 是相同的, 只是在文档方面有细微的变化, 并增加了一些说明。VRML 97 是 ISO/IEC 的标准。

### 2. VRML 形状

VRML 是由多个节点组成的层次结构, 可以描述由一个或多个对象构成的场景。VRML 包含一组基本的几何形状, 可以用这些基本形状构成更复杂的对象。Shape 节点是 VRML 中所有对象的通用节点。Box、Cylinder、Cone 和 Sphere 都是几何节点, 可以将基本对象放置在虚拟世界中。

VRML 支持定义包括 IndexedFaceSet 和 Extrusion 在内的复杂形状。IndexedFaceSet 是构成对象的一组表面。由于可以使用任意数目的表面, 所以支持创建复杂的形状。Extrusion 是一个沿着主轴突出的 2 维截面, 可以用来创建像花瓣那样的简单曲面。

对象的形状、大小、色彩和反射属性可以在 VRML 中规定。Appearance 节点控制形状的外观, 并可以包含一个 Material 节点和纹理节点。图 2-28 显示了其中一些形状。

Material 节点规定了对象的表面属性。它可以通过规定对象的红、绿、蓝值来控制对象的颜色。反射和发射的颜色也可以用类似的方法来定义。其他属性 (如对象反射直接和非直接光线的程度) 也可以控制。VRML 中的对象可以是透明或部分透明的。这些也包括在 Material 节点中。

有三类纹理节点可以用来将纹理匹配到任意对象上。最常用的是 ImageTexture, 它可以使用外部的 JPEG 或 PNG 图像文件并将其匹配到形状上。可以指定为对象匹配纹理的方法, 即图像倾斜到对象上的方法是可编辑的。

MovieTexture 节点允许将 MPEG 视频匹配到对象上, 还可以规定起始和结束时间。

最终的纹理匹配节点称为 PixelTexture, 表示使用 ImageTexture 来创建一个图像。尽管其效率不如 ImageTexture 节点, 但对于简单的纹理而言, 它仍然是很有用的。

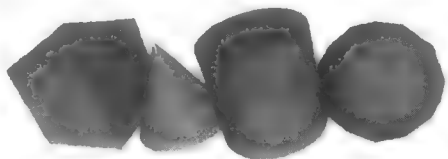


图 2-28 基本的 VRML 形状

51

52

文本可以通过 Text 节点加入到 VRML 中。你可以规定需要加入的文本,以及文本的字体、对齐方式以及大小等。在默认情况下,文字朝着 Y 轴的正方向,或者说“向上”。

所有的形状和文本都在 VRML 世界的中间开始。为了对形状进行排列,我们必须使用 Transform 节点来包装形状节点。Transform 节点可以包括 Translation、Scale 和 Rotation 节点。Translation 将对象从它的当前位置(默认为 VRML 世界的中心)移动一段特定的距离。Scale 增加或缩小对象的大小,Rotation 则使对象绕其中心旋转。

### 3. VRML 世界

一个虚拟的世界除了逼真的形状之外,还需要观察对象的摄像机以及背景和光照。默认的摄像机与 Z 轴的负方向对齐,和场景的中心相距数米。可以使用 Viewpoint 节点来改变默认的摄像机位置或增加新的摄像机。图 2-29 显示了一个从视角看到的 VRML 场景。



图 2-29 一个简单的 VRML 场景

我们可以使用 position 节点来指定视角,并用 orientation 节点从默认视角处进行旋转。摄像机可以通过 fieldOfView 节点来改变它相对于其投影区域的角度(默认值为 0.78 弧度)。改变投影区域角度可以获得从远处拍摄的效果。

在 VRML 世界中,可以使用三种光照类型。DirectionalLight 节点用于在整个世界中沿着某一方向进行照射,这和太阳光比较相似,太阳光来自一个方向并且影响场景中的所有对象。PointLight 则从空间一个固定点上向四周发射光线。SpotLight 则从一点沿着一个方向发射光线。合适的光照对于增加场景逼真度是非常重要的。对于任何一类光照,我们都可以为其规定很多参数,如光照的色彩和亮度。

VRML 世界的背景可以通过 Background 节点来指定。背景的颜色(默认为黑色),以及天空的颜色都可以加以修改。Panorama 节点可以将纹理和世界的侧面进行匹配。可以将一个全景覆盖到包围 VRML 世界的大立方体上。如果我们使用全景,那么用户将无法处理纹理,因为用户处于全景的中心。此外我们还可以使用 Fog 节点来为 VRML 增加烟雾的效果,用户可以指定雾的颜色和强度。雾可以增加世界的帧率,因为被雾覆盖的对象不会被渲染。

#### 2.3.2 动画和交互

VRML 97 与最初的 VRML 1.0 相比,一大优势是 VRML 世界可以是交互性的。VRML 中唯一的动画方式是渐变,这可以通过逐渐改变由内插节点规定的对象来实现。这个节点将会根据其内插类型逐渐改变对象。

内插类型有 5 种:色彩、坐标、普通、方向、位置和标量。所有的内插都有两个必须指定的节点: key 和 keyValue。key 由两个或多个数字的列表组成,从 0 开始到 1 结束。每一个 key 元素必须有 keyValue 元素作为补充说明。Key 定义了动画的进度,而 keyValue 则定义了应改变为什么值。例如,一个值为 0.5 的 key 元素以及匹配的 keyValue 定义了当动画进行到一半时的对象外观。

TimeSensor 节点为动画计时,这样内插节点就知道对象应该处于什么阶段。TimeSensor 在 VRML 世界中并没有物理形状,只是起到计时的功能。为了通知内插节点时间的变化,需要用 ROUTE 连接两个节点。TimeSensor 和内插节点之间以及内插节点和动画对象之间各需要一个 ROUTE。大多数动画都是通过这种方法完成的。改变 ROUTE 命令,使得一个事件能够触发其他

多个事件可以获得复杂的动画效果。

为了获得用户的输入，VRML 中使用两种类型的传感器。第一种为环境传感器。环境传感器有三种节点：VisibilitySensor、ProximitySensor 和 Collision。当用户的投影区域进入一个不可见的盒子时，VisibilitySensor 被激活。当用户进入或离开一个区域时，ProximitySensor 被激活。当用户装上节点时，Collision 节点被激活。

第二类传感器称为指向设备传感器。一种指向设备传感器是触摸传感器，当用鼠标激活该对象时激活触摸传感器。另外三种传感器称为拖拉传感器。这些传感器支持通过鼠标拖拉来旋转球体、柱体和平面。

### 2.3.3 VRML 规范

VRML 文件是具有.wrl 后缀的文本文件。VRML 97 在文件的第一行必须包括#VRML V2.0 UTF8 语句。#在文件第一行之外的其他位置表示注释。VRML 文件的第一行告诉 VRML 客户端所使用的 VRML 版本。VRML 的节点是区分大小写的，通常采用层次的形式构造。

54

尽管简单的文本编辑器（如记事本）就可以用来处理 VRML 文件，但仍然有一些 VRML 专用的编辑器，如 VRMLpad。它们可以通过提供不同的色彩和折叠（或是扩展）节点来帮助我们创建 VRML 对象。

所有的节点都用“{”开头，用“}”结尾，节点之间可以嵌套包含。名为组节点的特殊节点可以聚集多个节点。关键字 children 以及其后的“[”作为子节点的开头，并由“]”结尾。Transform 节点就是一个组节点。

可以用 DEF 来定义节点，用 USE 来重用节点。而且，可以应用多个简单对象创建复杂对象。

下面的语句在 VRML 中创建一个简单的盒子：

```
Shape {
  Geometry Box{}
}
```

盒子默认为位于屏幕中央的两米长的立方体。将它放入 Transform 节点可以使这个盒子移动到场景中的不同位置。我们也可以给盒子赋予不同的颜色，如红色：

```
Transform { translation 0 10 0 children [
  Shape {
    Geometry Box{}
    appearance Appearance {
      material Material {
        diffuseColor 1 0 0
      }
    }
  }
]
```

这个 VRML 片段将一个红色的盒子放置于+10Y 的方向。如果 DEF mybox 在 Transform 的前面，那么这个盒子就可以重用。现在，当盒子需要再次被使用时，只需要使用 USE mybox 就可以得到它的拷贝。

## 2.4 进一步探索

要初步了解多媒体编著，可以参考[3，1]和[4]中的第 5~8 章。而自动编著的详细介绍可以参考[7]。

本书的网站中第 2 章的 Further Exploration 提供了链接，可以获得有关多媒体编著的一个完整的 FAQ 文件。

图 2-5 显示的用来研究互补色的 TextColor.exe 程序也可以在本书的网站上获得。

55

本书网站包括了关于 Director 的 FAQ 的链接。展示 2.2.2 节主要思想的一个 Director 影片也可以从网站中下载, 此外还有关于 Dreamweaver 和 VRML 的信息, 以及 VRML 的一个小 demo。

## 2.5 练习

### 1. 多媒体还擅长表达哪类附加信息?

- (a) 有什么内容是口头文本可以表达而书面文本不能表达的?
- (b) 在什么情况下书面文本比口头文本更为理想?

### 2. 在你的实验室中找到并学习 3D Studio Max。阅读在线教程, 掌握软件的 3D 建模技术。学习如何使用它进行纹理匹配和制作动画。最后制作一个 3D 模型。

### 3. 用 Dreamweaver 设计一个交互式网页。HTML 4 中提供了类似 Adobe Photoshop 中的层的功能。每一层表示一个 HTML 对象, 如文本、图像或一个简单的 HTML 网页。在 Dreamweaver 中, 每一层都有一个与该层相关的符号。所以, 选中符号就可以选中整个层, 然后就可以进行进一步的处理。在 Flash 中, 你可以添加按钮和行为以便进行浏览和控制。你还可以使用时间轴行为来创建动画。

### 4. 考虑自动编著, 回答以下问题:

- (a) 你认为“活动图像”的含义是什么?
- (b) 如果将基于文本的技术移植到基于图像的自动编著中, 可能产生什么问题?
- (c) 使用现有的文本文档进行自动编著的最主要问题是什么?

56

### 5. 假设我们希望创建一个简单的动画, 如图 2-30 所示。注意, 这幅图像是动画在某一时刻的静态画面, 并非表示动画是鱼的移动过程, 在鱼的移动过程中它是重复的。说明为了实现这个目的我们需要做什么, 并给出解决这个问题的伪代码。

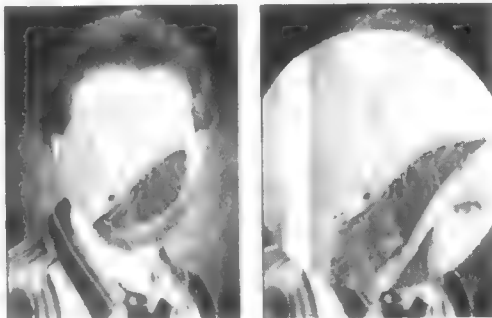
假设我们已经有了表示鱼的路径的坐标  $(x, y)$ , 我们可以调用一个过程来将图像定位在路径位置上, 并且运动是在视频的顶部发生。

### 6. 对于图 2-11 中的滑动切换, 解释我们如何得到非运动的右边的视频 $R_R$ 中 $x$ 的表达式。

### 7. 假设我们希望实现一种视频切换方式: 第二段视频将第一段视频覆盖, 并通过一个开放的圆圈慢慢显现(类似于相机的光圈), 如图 2-31 所示。写出公式, 从两段视频中获得正确的像素点以实现这种功能。只需写出红色通道的答案即可。



图 2-30 分镜, 渐渐占据更多的空间



a) 光圈打开

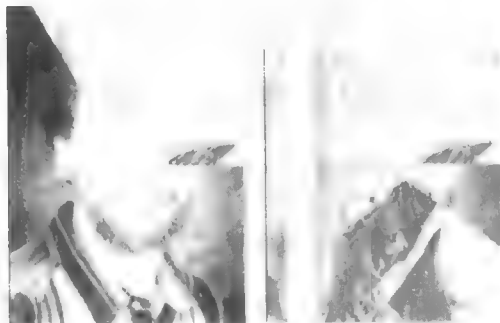
b) 稍后

图 2-31 虹膜擦拭

57

### 8. 现在假设我们希望实现一种视频切换方式, 第二段视频仍将第一段视频覆盖, 并通过一条运动的半径慢慢显现(如钟表的指针), 如图 2-32 所示。写出公式, 从两段视频中获得正确的像素点以实现这种功能。只需写出红色通道的答案即可。

9. 假设你希望实现一种波动的效果, 如图 2-33 所示。用与  $x$  有一定偏移处的值来代替  $x$  处的值就可以得到这种效果。假设图像的大小为 160 行×120 列。



a) 钟表的指针正在掠过

b) 稍后

图 2-32 钟表掠过

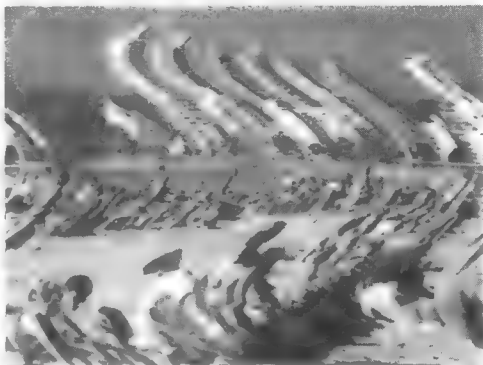


图 2-33 视频中的滤镜

- (a) 使用浮点运算在像素点  $x$  的值上增加一个正弦分量, 使得  $x$  的 RGB 值等于原图中另一个像素点的值。 $x$  的最大切换值是 16。
- (b) 在 Premiere 以及其他软件包中, 只提供了整数运算。像正弦这样的函数都要重新定义, 它只接受整数形式的参数并且返回整数形式的值。 $\sin$  函数的参数值必须是 0~1024, 而其值为 -512~512, 即  $\sin(0)$  返回 0,  $\sin(256)$  返回 512,  $\sin(512)$  返回 0,  $\sin(768)$  返回 -512,  $\sin(1024)$  返回 0。

对于 (a) 的问题, 使用整数运算重写表达式。

- (c) 如果要使波动与时间相关, 如何修改答案?

10. 如何创建图 2-6 所示的图像? 编写一个小程序用以创建这样的图像。提示: 将 R、G、B 分别放置在圆的内接等边三角形的顶点上。最好能对输出对象的行和列进行检查, 而非简单地将结果按照  $(x, y)$  的位置进行匹配。
11. 为了学习如何使用现有的软件对图像、视频和音乐进行操作, 制作一段时长 1 分钟的数字视频。通过这个练习, 你将熟悉如何使用基于 PC 的设备以及 Adobe Premiere、Photoshop、Cakewalk Pro Audio 等多媒体软件。
  - (a) 捕获 (或寻找) 至少三个视频文件。你可以使用摄像机或 VCR 制作自己的 (使用 Premiere 或类似软件) 视频文件或在网上搜寻。
  - (b) 使用 Cakewalk Pro Audio 创作 (或编辑) 一个小的 MIDI 文件。
  - (c) 创建 (或寻找) 至少一个 WAV 文件。你可以自己创作或从网上下载。
  - (d) 使用 Photoshop 创建一个标题和一个结尾。
  - (e) 将上面各步的结果组合起来, 创作一段 60 秒左右的影片, 包括标题、创作人员、配乐和至少三处切换。试验不同的压缩方法, 最终版本最好使用 MPEG。
  - (f) 以上部分构成了实验的最基本部分。你可以进一步地改进你的作品, 不过千万不要因此而废寝忘食啊!

58

## 2.6 参考文献

- [1] A.C. Luther, *Authoring Interactive Multimedia*, The IBM Tools Series, San Diego: AP Professional, 1994.

- [2] R. Vetter, C. Ward, and S. Shapiro, "Using Color and Text in Multimedia Projections," *IEEE Multimedia*, 2(4): 46–54, 1995.
- [3] J.C. Shepherd and D. Colaizzi, *Authoring Authorware: A Practical Guide*, Upper Saddle River, NJ: Prentice Hall, 1998.
- [4] D.E. Wolfgram, *Creating Multimedia Presentations*, Indianapolis: Que Publishing, 1994.
- [5] A. Ginige, D. Lowe, and J. Robertson, "Hypermedia Authoring," *IEEE Multimedia*, 2: 24–35, 1995.
- [6] A. Ginige and D. Lowe, "Next Generation Hypermedia Authoring Systems," In *Proceedings of Multimedia Information Systems and Hypermedia*, 1995, 1–11.
- [7] D. Lowe and W. Hall, *Hypermedia and the Web: An Engineering Approach*, New York: Wiley, 1999.



## 第3章 图形和图像的数据表现

在本章中，我们将介绍图像，先介绍 1 位图像，然后介绍 8 位灰度图像，以及怎样绘制它们，接着介绍 24 位彩色图和 8 位彩色图。此外，还会讨论一些用于存储这种图像的文件格式的细节。

我们将涉及如下主题：

- 图形/图像的数据类型。
- 常用的文件格式。

### 3.1 图形/图像的数据类型

用于多媒体的文件格式的数量不断增长[1]。比如，表 3-1 显示了 Macromedia Director 中可以使用的文件格式的列表。在本书中，我们将只介绍一部分常用的文件格式，并能解释它们怎样工作。我们将重点关注 GIF 和 JPG 文件格式，因为这两种文件格式能够被多数 Web 浏览器解压缩和显示。

首先，我们将从总体上讨论文件格式的特性。

表 3-1 Macromedia Director 的文件格式

文件导入					文件导出		本地
图像	调色板	声音	视频	动画	图像	视频	
BMP、DIB、	PAL	AIFF	AVI	DIR	BMP	AVI	DIR
GIF、JPG、	ACT	AU	MOV	FLA		MOV	DXR
PICT、PNG、		MP3		FLC			EXE
PNT、PSD、		WAV		FLI			
TGA、TIFF、				GIF			
WMF				PPT			

#### 3.1.1 1 位图像

图像由像素（pixel）组成，像素是数字图像中的图片元素。一幅 1 位图像仅仅由“开”位和“关”位组成，这就是最简单的图像格式。图像的每个像素作为一个位存储（0 或者是 1）。因此，这样一幅图像称为二值图像（binary image）。

二值图像也称为 1 位单色（monochrome）图像，因为它不包含其他颜色。图 3-1 显示了一幅 1 位单色图像（多媒体科学家称为“Lena”，这是一幅标准图像，用来说明许多算法）。一幅分辨率为 640×480 的单色图像需要 38.4KB（即 640×480/8）的存储空间。1 位单色图像适用于仅含有简单图形和文本的图片。



图 3-1 单色 1 位 Lena 图像

#### 3.1.2 8 位灰度图像

现在来考虑 8 位图像，即图像的每个像素有一个在 0~255 之间的灰度值（gray value）。每个像素由一个字节表示。例如，一个暗的像素可能具有灰度值 10，而一个亮的像素可能具有灰度值 230。

整幅图像可以看作由像素值组成的二维数组。我们称这样一个二维数组为一幅位图（bitmap），

即图形/图像的数据表现。它与图形/图像在视频内存中的存储方法类似。

图像分辨率 (image resolution) 指的是数字图像中的像素数目 (分辨率越高则图像质量越好)。对一幅图像, 非常高的分辨率可能是  $1600 \times 1200$ , 而低一点的分辨率可能是  $640 \times 480$ 。注意, 这里我们用的屏幕长宽比是 4:3。我们不一定非要用这个比例, 但是用这个比例看起来更自然。

这样的二维数组必须存储在硬件中, 我们称这样的硬件设备为帧缓存 (frame buffer)。被称为“视频”卡 (实际上是图形卡) 的专门的硬件设备 (相对较昂贵) 就是用于这个目的。视频卡的分辨率不必达到图像所要求的分辨率, 但是如果视频卡没有足够的存储空间, 为了显示, 数据就不得不在 RAM 中移动。

我们可以把 8 位图像看作一组 1 位位平面 (bitplane), 其中每个平面由图像的 1 位表示组成, 而平面的“海拔”不断升高, 即如果图像的像素在某个位的值不为 0, 则这个位被置为 1。

61

图 3-2 用图显示了位平面的概念。每个位平面的每个像素具有 0 或 1 值。但是, 所有的位平面的对应像素的位组成一个字节, 用来存储 0~255 间 (8 位情形下) 的某个值。对于影响最小的位, 在二进制数的最终数字和中, 该位的值变为 0 或者 1。位置算法隐含着对于下一个 (即第二个) 位, 该位的 0 或 1 对最终和的贡献是 0 或 2。依次下去的位代表 0 或 4, 0 或 8, 等等, 直到影响最大的位贡献值为 0 或 128。视频卡能够以视频速率刷新位平面的数据, 但是不像 RAM 那样能很好的保持数据。在北美, 光栅场以每秒 60 个周期的速度刷新。

每个像素通常用一个字节存储 (值从 0~255), 所以一幅  $640 \times 480$  的灰度图需要 300KB 的存储空间 ( $640 \times 480 = 307\ 200$ )。图 3-3 再次显示了 Lena 图像, 这次是一幅灰度图。

如果我们想打印 (print) 这样一幅图, 那么事情将变得复杂一些。假设我们有一个 600dpi 的激光打印机。这样的设备通常只能打印一个点或者不打印一个点。然而, 将一幅  $600 \times 600$  的图像打印在一平方英寸的空间内, 其结果就不能令人十分满意。此时采用抖动 (dithering, 也被称为抖色) 方法。它的基本策略是以亮度分辨率换取空间分辨率。(见[2]的 568 页, 该书对抖动有更好的讨论。)

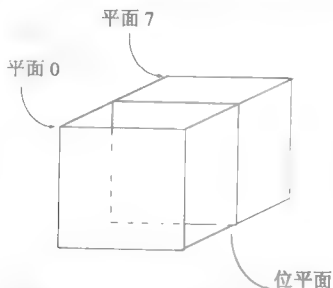


图 3-2 8 位灰度图像的位平面



图 3-3 Lena 灰度图

### 抖动

对于在 1 位打印机上的打印来说, 用抖动来计算出更大模式的点, 这样, 0~255 之间的值就与能够正确表现明暗的像素值的模式相对应。主要的策略是用一个更大的模式 (例如  $2 \times 2$  或  $4 \times 4$ ) 替代一个像素值, 这样, 被打印点的数量近似于用在网板打印 (halftone printing) 的油墨的变化大小的圆盘字模式。网板打印是一种用于报纸印刷的模拟过程, 用或小或大的黑油墨实心圆来表现阴影。

如果我们用  $n \times n$  的开关 1 位点矩阵来代替, 我们就能表现  $n^2+1$  级亮度分辨率。比如, 矩阵中任意三点被打印成黑色可算作一个亮度级。点模式用试探法创建。比如, 我们用一个  $2 \times 2$  的“抖动矩阵”:

62

$$\begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$$

我们可以先用除数是 256/5 的整数除法把 0~255 间的图像的值映射到一个新的范围 0~4。接下来,比如,如果像素值是 0,那么在打印机的一个 2×2 输出区域不进行打印;但如果像素值是 4,则我们在 4 个点都填涂。所以规则如下:

如果像素亮度比抖动矩阵的某个元素的编号大,则在该元素点填涂,即用一个  $n \times n$  的点矩阵替代每个像素。

然而,我们注意到上述这种打印方式的亮度级数量还太小。而如果我们通过增加抖动矩阵的大小来增加亮度级数量,我们也会增加输出图像的大小。这就降低了图像在任何局部的清晰度,从而降低了图像的空间分辨率。

注意,对于一个经抖动方法处理的图像,其尺寸可能太大,因为就算用 4×4 矩阵替代每个像素点,也会使图像变为原来的 16 倍。然而,利用“有序抖动”的方法可以解决这个问题。假设我们希望用一个更大的 4×4 的抖动矩阵,如

$$\begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix}$$

然后假设我们把这个抖动矩阵在同一时刻移动到图像在水平和垂直方向的四个像素上(这里的图像值已经被降低到 0~16 这个区间中)。如果某个像素的亮度值大于覆盖它的矩阵元素的编号,则在打印机的相应元素输出位填涂。图 3-4a 显示了 Lena 的灰度图。有序抖动版本在图 3-4b 中显示,其中 Lena 右眼的详细情况在 3-4c 中显示。

63



图 3-4 灰度图像的抖动

一个使用  $n \times n$  抖动矩阵的用于有序抖动的算法如下:

算法 3-1 有序抖动

```

begin
  for x = 0 to xmax           // columns
    for y = 0 to ymax         // rows
      i = x mod n
      j = y mod n
      // I(x, y) is the input, O(x, y) is the output, D is the dither matrix.
      if I(x, y) > D(i, j)
        O(x, y) = 1;
      else
        O(x, y) = 0;
    end
  end

```

Foley 等的著作[2]对有序抖动进行了更详细的介绍。

### 3.1.3 图像数据类型

本节介绍图形和图像文件格式的一些最常见的数据类型：24 位色和 8 位色。然后我们将讨论文件格式。一些文件格式只能用于特殊的硬件和操作系统平台，而另一些格式是平台无关（platform-independent）或者说是跨平台（cross-platform）的文件格式。即使有些格式不是跨平台的，但转换程序能识别这些格式并将其转换成其他格式。

由于图像文件需要较大的存储空间，因此大多数图像格式都结合了某种压缩（compression）技术。压缩技术可划分为无损（lossless）压缩和有损（lossy）压缩。我们将在第 7 章~第 14 章学习各种图像、视频和音频压缩技术。

### 3.1.4 24 位彩色图像

64 在一个 24 位彩色图像中，每个像素用三个字节表示，通常表示为 RGB。因为每个值的范围是 0~255，这种格式支持  $256 \times 256 \times 256$ （总共 16 777 216）种可能的颜色组合。然而，这种灵活性会导致存储问题。例如，一幅  $640 \times 480$  的 24 位彩色图像如果不经压缩，需要 921.6KB 的存储空间。

要注意的一个重要问题是，许多 24 位彩色图像通常存储为 32 位图像，每个像素多余的数据字节存储一个  $\alpha$ （alpha）值来表现有特殊影响的信息。（见[2]的 835 页，介绍使用  $\alpha$  通道在图形图像中合成一些重叠物体。最简单的使用就是用来作为一个透明标签。）

图 3-5 显示了图像 forestfire.bmp，一幅用 Microsoft Windows BMP 格式（本章后面部分讨论）显示的 24 位图像。这幅图像的红、绿、蓝通道的灰度图分别在 3-5b、3-5c、3-5d 中显示。每个颜色通道的 0~255 的字节值表示亮度，我们就能够用每种颜色分别显示灰度图。

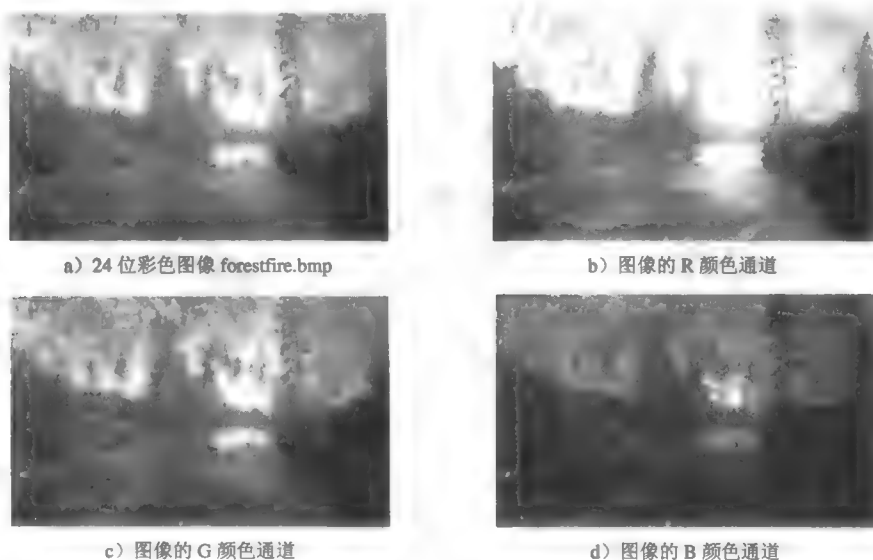


图 3-5 高分辨率的彩色图像和单独的 R、G、B 颜色通道图像。（彩色插页中也有此图）

### 3.1.5 8 位彩色图像

65 如果空间相关（实际上几乎总是这样），通过量化颜色信息就可以得到足够精确的彩色图像。许多系统只能利用 8 位颜色信息（也称作 256 色）来生成屏幕图像。即使有些系统有能真正使用 24 位颜色信息的电路，向后兼容的特性也要求我们能理解 8 位彩色图像文件。

这样的图像文件使用了查找表(lookup table)概念来存储颜色信息。基本上,图像存储的不是颜色而仅仅是字节的集合,每个字节是指向一个表的索引。该表表项具有三字节值,指明了像素(带有查找表索引)的颜色。从某种程度上讲,它有点像小孩的按序号画图的绘画册,数字1可能代表橙色,数字2代表绿色等——真正的颜色集不一定是这个模式。

仔细选择在一幅图中使用哪些颜色具有最佳表现力是有意义的:如果一幅图像描绘的是日落场景,那么精确地表现红色而仅存储少量的绿色是合理的。

假设一个24位图像中所有的颜色都放在一个 $256 \times 256 \times 256$ 的格子集里,同时格子中,还要存储每种颜色对应的像素数目。比如,如果23个像素具有RGB值(45, 200, 91),那么把23存储在一个三维数组中,元素下标值为[45, 200, 91]。这种数据结构被称为颜色直方图(color histogram)。(见[3][4]。)

图3-6显示了forestfire.bmp中像素的RGB值的三维直方图。直方图有 $16 \times 16 \times 16$ 个小格,并用亮度和伪彩色的形式显示了每个小格中的像素数目。我们能够看到一些颜色信息的重要的聚类,对应forestfire图像的红色、绿色、黄色等。这样的聚类方法让我们能够选出最重要的256种颜色。

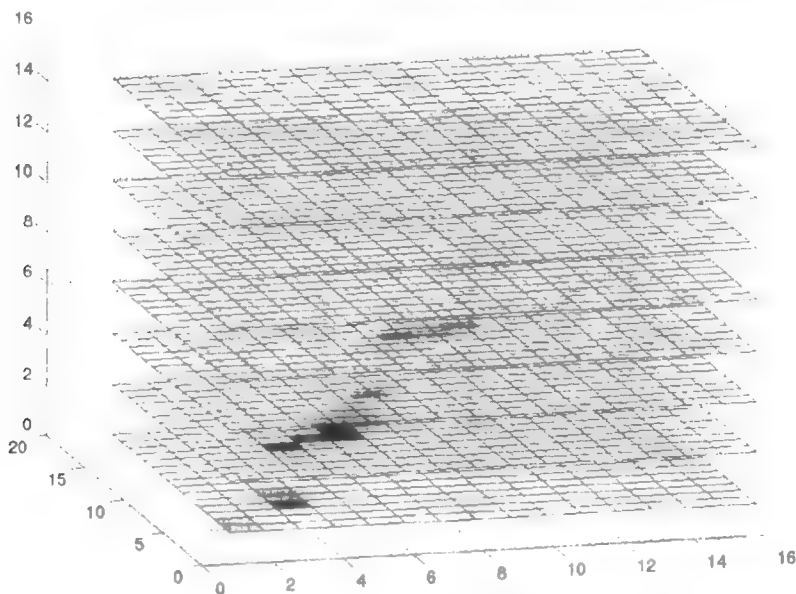


图3-6 在forestfire.bmp中的RGB色彩的三维直方图

基本上,三维直方图小格中的大数据量能够由分割-合并(split-and-merge)算法解决,以选取最好的256种颜色。图3-7显示了作为结果的GIF格式(本章后面部分讨论)的8位图像。注意,很难区分图3-5a显示的24位图像和图3-7显示的8位图像的差别。但情况并不总是这样。考虑一下医学图像的情况:你是否对仅仅是“合理精确”的用于激光外科的你的大脑图像满意呢?可能不会吧。那就是为什么要考虑在医学应用中使用64位图像的原因。



图3-7 8位彩色图像的例子  
(彩色插面中也有此图)

请注意，8 位图像比 24 位图像大大节省空间，一个 640×480 的 8 位彩色图像只需要 300KB 的容量，而真彩色图像就需要 921.6KB 的空间（在没用任何压缩的情况下）。

### 3.1.6 颜色查找表

回顾一下，在 8 位彩色图像使用的理念是仅仅为每个像素存储下标，或者说是编号值。那么，如果一个像素存储的值是 25，意为在颜色查找表（Lookup Table, LUT）中找到第 25 行。因为图像用二维数组值显示，故它们通常按照行—列顺序存储为一长串的值。对一幅 8 位图像，图像文件可以在文件头信息中存储每个下标对应的 8 位 RGB 值。图 3-8 显示了这种方法。颜色查找表通常被称为调色板（palette）。

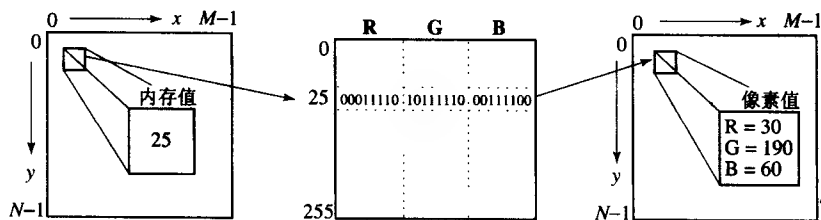


图 3-8 8 位彩色图的颜色查找表

一个颜色选择器（color picker）由比较大的颜色块（或者说是断续的颜色区间）的数组组成，这样鼠标点击就能够选中指向的颜色。实际上，一个颜色选择器显示其下标值为 0~255 的调色板颜色。图 3-9 显示了颜色选择器的概念。如果用户选择了下标值为 2 的颜色块，那么该颜色为青色，RGB 值为（0，255，255）。

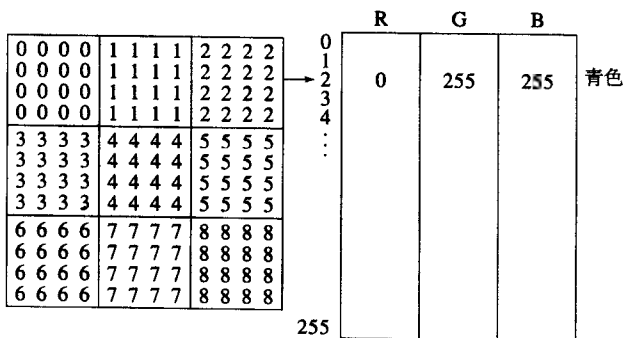


图 3-9 8 位颜色的颜色选择器：颜色选择器的每一块对应颜色查找表的每行

通过修改颜色表就可能制作简单的动画，这称为色彩循环（color cycling）或调色板动画（palette animation）。因为从颜色表更新非常快，这样会获得简单但令人满意的效果。

抖动技术同样可以用来进行彩色打印。每个颜色通道用 1 位，同时用 RGB 点把颜色隔开。作为选择，如果打印机或者屏幕只能打印有限数量的颜色，比如用 8 位而不是 24 位，那么即使在颜色查找表中无法找到的颜色也是可以打印的。通过平均相邻像素点的亮度，显示的颜色分辨率可以明显增加而不用降低空间分辨率。那么，我们就能欺骗眼睛，让它感觉能看到不可获得的颜色，因为采用了空间混色。图 3-10a 显示了一幅 24 位彩色 Lena 图像；图 3-10b 显示了同一幅图通过抖动技术减少到 5 位的情况；图 3-10c 显示了左眼的细节。

怎样设计颜色查找表

在 3.1.5 节中，我们简单讨论了聚类（clustering）的思想，用来从一幅 24 位彩色图像中产生

最重要的 256 色。但是一般说来，聚类是代价昂贵又费时的办法。但我们的确需要设计颜色查找表，那么应该怎样做呢？



图 3-10

最直接的方法是在每一维把 RGB 立方体分成大小相等的块，然后将每个结果立方体中央的颜色作为颜色查找表的表项，只需要把 RGB 从 0~255 区间变为合适的区间就可以产生 8 位编号。

68

因为比起蓝色，人类对红色、绿色更敏感，所以我们可以将红色和绿色的范围从 0~255 缩小到 3 位 (0~7)，而蓝色的范围缩小到 2 位 (0~3，总共 8 位)。为了缩小红色和绿色区间，我们只需要用 32 (256/8) 除红色或是绿色的字节值，并截取结果的小数部分。这样图像中每个像素用 8 位下标代替，而颜色查找表提供了 24 位颜色。

然而，应用这种简单方案的后果是图像中会出现边缘效果。因为如果 RGB 中的轻微变化会导致移动到一个新的编号，边缘就出现了，这看起来令人很不舒服。

解决这种色彩减退问题的一个简单办法是采用中值区分算法 (median-cut algorithm)，一些其他方法也具有同样的效果或者是更好的效果。这种方法从计算机图形学[5]衍生而来；这里，我们仅介绍一个相当精简的版本。这个方法是自适应分区方案的一种，它试图表达最多的位，具有最强的识别能力，而颜色是最聚集的。

该算法思想是将红色字节的值排序，并找到其中值。然后，比中值小的值被标记为 0 位，比中值大的值被标记为 1 位。中值是一半像素值比其小，一半像素值比其大的点。

假设我们把一些苹果画成图，大多数像素都是红色的，那么红色字节值的中值可能落在 0~255 范围内的较高部分。下一步，我们仅仅考虑第一步后被标记为 0 的像素，并将它们的绿色值排序。接着，我们用另一位标记图像像素，0 标记绿色中那些比中值小的像素，1 标记那些比中值大的像素。然后，把同样的规则用于第一步后被标记为 1 的像素，这样我们给所有的像素点加上了 2 位标签。

继续在蓝色分量执行上述步骤，我们就得到了 3 位的方案。重复前面所有步骤，得到 6 位方案，再重复红色、绿色步骤，得到了 8 位方案。这些位形成了像素的 8 位颜色下标值，相应的 24 位颜色是最终小颜色立方体的中心。

可以看到，事实上这种方案关注那些最需要与大量颜色区别的位。利用显示 0~255 数目的直方图，我们能很容易地找到中值。图 3-11 显示了图 forestfire.bmp 的红色字节值的直方图，以及这些值的中值，用竖线表示。

69

用相应的颜色查找表中的 24 位色替代每个像素得到的 24 位彩色图像仅是原图的一个近似。当然，上述算法做了合理的工作：在最需要的地方（最需要注意小的颜色阴影差别）赋予了最大的识别能力。还需要提到的是，还有一些方法可用来把近似误差从一个像素分配到另一个。这具有平滑 8 位近似中的问题的效果。

中值区分算法的更精确的版本如下所示：

1) 找出最小的方形区，它包含图像中的所有颜色。

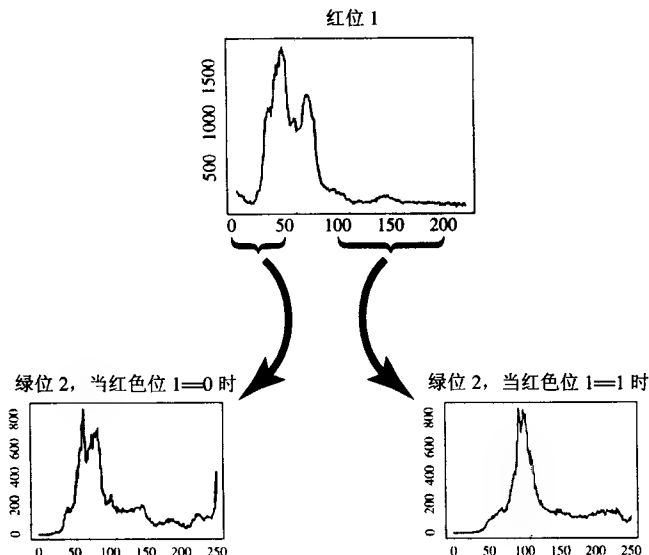


图 3-11 24 位彩色图像 forestfire.bmp 的红色字节直方图将每个像素标为 0 或者 1。对于正在构造的颜色表下标的第二位，我们认为 R 值比 R 的中值小，将这些像素标记 0 或者 1 是取决于它们的 G 值是比 G 的中值小或者大。继续直到覆盖 8 位的 RGB，最终给出颜色查找表的 8 位下标

2) 沿方形区的长边排序它所包含的颜色。

3) 在排序链表的中间处把该方形区划分成两个区域。

4) 重复上面步骤 2、3，直到初始的颜色空间被分割成 256 个区域为止。

5) 对每个方形区，把该方形区中 R、G、B 的平均值作为其代表（中心）颜色。

6) 根据一个像素的 RGB 值与每个方形区的中心值的欧氏距离，给每个像素分配一个代表颜色。在指向代表颜色的查找表中（其中，每种代表颜色是 24 位的，R、G、B 均为 8 位）用编号替代像素。

这样，我们就有了一个 256 行的表，每行含有 3 个 8 位值。行的下标是查找表的编号，这些下标正是存储在新的量化颜色（调色板）图像中的像素值。

## 3.2 常见的文件格式

下面介绍用于信息交换的一些流行的文件格式，其中最重要的一种是 8 位 GIF 格式，因为它与 WWW 和 HTML 标记语言有着历史渊源，它是能被网络浏览器识别的第一种图像类型。然而，目前最重要和常用的文件格式是 JPEG，它将在第 9 章中详细介绍。

### 3.2.1 GIF

图形交换格式（Graphics Interchange Format，GIF）是由 UNISYS 公司和 Compuserve 公司开发的，最初用于通过调制解调器在电话线上传送图形图像。GIF 标准采用了 Lempel-Ziv-Welch 算法（一种压缩格式，见第 7 章），但对该算法稍做改动，把图像扫描线包改造成有效地使用像素的线组。

GIF 标准仅适用于 8 位（256）彩色图像。因为这能够生成可接受的颜色，它最适合用于具有少量独特色彩的图像（如图形和绘画）。

GIF 图像格式有一些有趣的特点，尽管它在很多地方已经被取代了。该标准支持隔行扫描——通过套色（four-pass）显示方法处理，相隔的像素可以连续显示。

事实上，GIF 有两个版本，最初的规范是 GIF87a。后一个版本 GIF89a 通过数据中的图形控



制扩展 (Graphics Control Extension) 块支持简单的动画。它对延时、透明索引等提供了简单的控制。像 Corel Draw 这样的软件支持对 GIF 图像的访问和修改。

来看看支持 GIF87 的文件格式的一些细节是有意义的, 因为许多这样的格式与其有共同之处, 不过与这个“简单”标准相比, 已经变得复杂很多。对于这个标准规范而言, 常用的文件格式如图 3-12 所示。其中签名是 6 个字节: GIF87a; 屏幕描述符是 7 字节标志位。一个 GIF87 文件可以包含不止一种图像定义, 通常与屏幕不同的部分匹配。因此每个图像可以有颜色查找表, 即局部色图 (Local Color Map), 用于把 8 位映射成 24 位 RGB 值。然而, 局部色图不是必须的, 可以定义一个全局的色图并替代它。

屏幕描述符包含了属于文件中每个图像的属性。根据 GIF87 标准, 它在图 3-13 中定义。屏幕宽度在前两个字节给出。因为一些机器可能会转换 MSB/LSB (Most Significant Byte/Least Significant Byte) 顺序, 这个顺序就被指定了。屏幕高度在后两个字节。如果没有给出全局色图, 则将第 5 个字节的“m”置 0。颜色分辨率“cr”为 3 位, 范围从 0~7。因为这是一个老的标准, 通常在一些低端硬件上操作, “cr”需要很高的颜色分辨率。

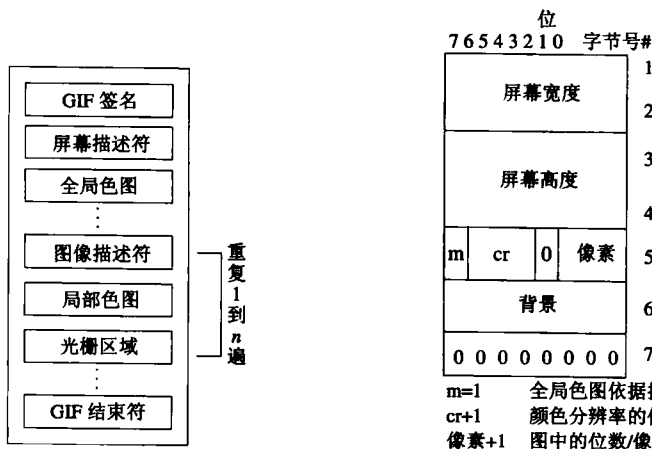


图 3-12 GIF 文件格式

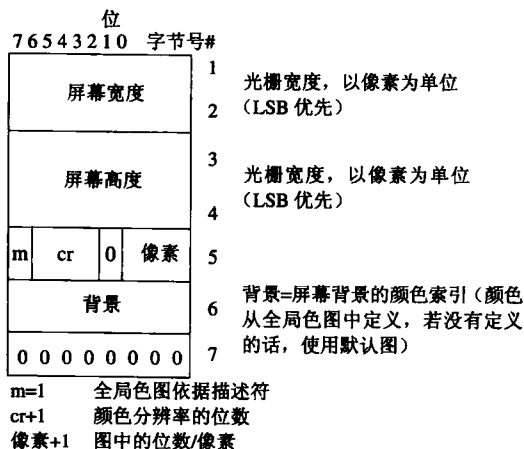


图 3-13 GIF 屏幕描述符

下一位显示为 0, 在该标准中没有用到。“像素”为另外 3 位, 指明图像文件中每个像素的位数。但是通常“cr”等于“pixel”, 故“pixel”不是必须的。第 6 字节给出背景颜色在颜色表中的下标, 第 7 字节填 0。在现在的使用中, 使用较小的颜色分辨率比较好, 因为我们可能会对非常低端的设备感兴趣, 像支持上网功能的手表。

可以用一个简单的方法建立色图 (color map), 如图 3-14 所示。然而, 正如屏幕描述符指出的, 表的真实长度等于  $2^{\text{pixel}+1}$ 。

文件中的每幅图像有自己的图像描述符, 在图 3-15 中定义。有趣的是, 标准的制定者们为了将来的扩展, 忽略了一幅图像末尾和另一幅图像开始的一些字节, 用逗号来识别。这样, 将来的扩展就非常容易做到向后兼容。

如果局部图像描述符中的隔行位 (interlace) 被设置, 则图像的行以 4 次分别套色顺序显示, 如图 3-16 所示。这里, 第一个通道显示第 0 和第 8 行, 第二个通道显示第 4 和第 12 行, 依此类推。这样, 当一个 Web 浏览器显示图像

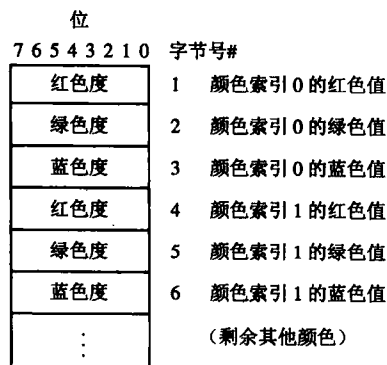


图 3-14 GIF 颜色映射

时，就能让图像概略迅速显示，进而再显示详细细节。下面的 JPEG 标准具有相似的显示方式，这种方式称为渐进式（progressive）方式。

位								字节号#	
7	6	5	4	3	2	1	0		
0	0	1	0	1	1	0	0	1	图像分隔符字符（逗号）
图像的左边								2	图像在屏幕左边的开始处，以像素计（LSB 优先）
								3	
图像的上边								4	图像在屏幕上边的开始处，以像素计（LSB 优先）
								5	
图像的宽度								6	图像的宽度，以像素计（LSB 优先）
								7	
图像的高度								8	图像的高度，以像素计（LSB 优先）
								9	
m	i	0	0	0	0	0	像素	10	m=0 使用全局色图，忽略“像素” m=1 局部色图遵循，使用“像素” i=0 以顺序序格式化图像 i=1 以交错序格式化图像 像素+1 此图像每像素的位数

图 3-15 GIF 图像描述符

图像行	扫描 1	扫描 2	扫描 3	扫描 4	结果
0	*1a*				*1a*
1				*4a*	*4a*
2			*3a*		*3a*
3				*4b*	*4b*
4		*2a*			*2a*
5				*4c*	*4c*
6			*3b*		*3b*
7				*4d*	*4d*
8	*1b*				*1b*
9				*4e*	*4e*
10			*3c*		*3c*
11				*4f*	*4f*
12		*2b*			*2b*
⋮					

图 3-16 GIF 套色隔行扫描显示行顺序

真实光栅数据（actual raster data）本身在存储前先用 LZW 压缩方案（见第 7 章）压缩。

为了将来的使用，GIF87 标准也设定了扩展块该怎样定义。即使在 GIF87 中，也可以获得简单的动画，但是没有定义图像间的延迟。多幅图像只会一幅幅地覆盖，而不会清屏。

GIF89 引进了许多扩展块定义，特别是那些和动画相关的定义，如透明和图像间延迟。GIF89 中引入的一个相当有用的特性是排序颜色表的思想。最重要的颜色率先出现，这样，如果解码器只能使用较少的颜色，那么最重要的颜色就被选中了。也就是说，仅使用颜色查找表中的一部分，邻近的颜色尽可能映射到可利用的颜色。

我们通过看一个 GIF 图像，就可了解文件头是怎样工作的。图 3-7 是一幅 8 位彩色 GIF 图像。为了了解文件头，我们可以使用 UNIX 操作系统中比较常用的命令：od（octal dump）。在 UNIX<sup>Ⓔ</sup> 下，我们使用命令

Ⓔ Solaris 版本，老的版本的语法会有所不同。

73  
74

```
od -c forestfire.gif | head -2
```

这样，我们就可以看到前 32 个字节被解释为如下字符：

```
G I F 8 7 a \208 \2 \188 \1 \247 \0 \0 \6 \3 \5
J \132 \24 | ) \7 \198 \195 \ \128 U \27 \196 \166 & T
```

为了解释文件头的剩余部分（GIF87a 之后的部分），我们使用 16 进制：

```
od -x forestfire.gif | head -2
```

得到结果：

```
4749 4638 3761 d002 bc01 f700 0006 0305
ae84 187c 2907 c6c3 5c80 551b c4a6 2654
```

其中，签名后的 d002 和 bc01 是屏幕宽度和高度。它们以低字节在前的顺序表示，所以对这个文件，屏幕宽度用 10 进制表示是  $0+13\times 16+2\times 16^2=720$ ，屏幕高度是  $12+11\times 16+1\times 16^2=444$ 。然后，f7（10 进制的 247）是屏幕描述符的第 5 个字节，之后是背景色索引 00 和分隔符 00。标志符 f7 以位形式表示是 1,111,0,111。换句话说，采用全局色图，8 位颜色分辨率，0 分隔符，8 位像素数据。

### 3.2.2 JPEG

目前最重要的图像压缩标准是 JPEG[6]。该标准由国际标准化组织（International Organization for Standardization, ISO）的一个工作组制定，该工作组的一个非正式称呼为联合图像专家组（Joint Photographic Experts Group），JPEG 也因此而得名。我们将在第 9 章详细学习 JPEG，不过这里先介绍它的一些主要特点。

JPEG 利用了人类视觉系统一些特定的局限性，从而获得很高的压缩率。眼-脑系统看不到极为精准的细节。如果一些像素发生了许多变化，我们称图像的那个部分具有高空间频率，即在  $(x, y)$  空间发生了许多变化。和灰度图相比，这个局限对于彩色图更为明显。因此，JPEG 中的颜色信息被大量丢掉并且图像中的小块用空间频率域  $(u, v)$  来表现，而不是用  $(x, y)$ ，即  $x$  和  $y$  的改变速度被从低到高进行估计，然后通过将这些速度的系数或权值进行分组，形成一个新的“图像”。

人们总是喜欢缓慢变化的权值，这里有一个简单的技巧：值被大的整数除然后截尾。这样，小的值就变成了零。然后采用能有效表示一长串 0 的方案。瞧！图像被极大的压缩了。

因为我们在除和截短步骤中扔掉了许多信息，这种压缩方案是有损的（尽管也有了无损的方式）。而且，因为让用户直接选择使用多大的分母也就是决定了多少信息将丢失，JPEG 允许用户设定需要的质量等级，或者说是压缩率（输入除以输出）。

作为一个例子，图 3-17 展示了 forestfire 图像，质量因数  $Q=10\%$ （通常默认的质量因数为  $Q=75\%$ ）。

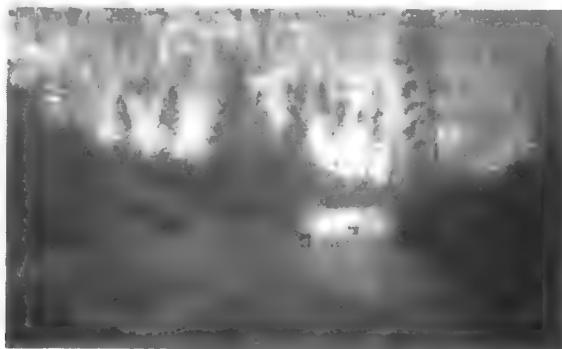


图 3-17 用户指定的低质量的 JPEG 图像（彩色插页中也有此图）

这幅图仅是原图大小的 1.5%。在压缩时，一幅  $Q=75\%$  的 JPEG 图像成了大小为原图的 5.6% 的图，而这幅图像的 GIF 版本大小压缩到原图的 23.0%。

### 3.2.3 PNG

随着因特网的流行，研究人员花费大量的精力研究更多与系统无关的图像格式。这样的一个是便携式网络图形 (Portable Network Graphics, PNG)。这个标准有望取代 GIF 标准并做了重要扩展。提出一个新标准的原因部分在于 UNISYS 和 Compuserve 在 LZW 压缩方法上的专利。(有趣的是，该专利仅仅有压缩，而没有解压缩，这也是为什么 UNIX 的 gunzip 功能可以将 LZW 压缩文件解压缩的原因。)

PNG 的独特之处是最多可支持 48 位的色彩信息——这是非常大的进步。文件可能还包含用于正确显示彩色图像的伽马校正信息 (见 4.1.6 节) 以及用于透明控制的阿尔法通道信息。与 GIF 基于宽展行的渐进显示不同，PNG 的显示在 7 通道上对图像的每个  $8 \times 8$  块一次同时在二维空间显示一些像素。

### 3.2.4 TIFF

标记图像文件格式 (Tagged Image File Format, TIFF) 是另一种常用的文件格式。它由 Aldus 公司在 20 世纪 80 年代开发，后来受到 Microsoft 的支持。它为支持附带额外信息 (被称为标记) 提供了许多方便。最重要的标记就是格式表示 (format signifier)，即在图像中用了哪些压缩类型，等等。比如，TIFF 可以存储不同的图像类型，包括 1 位、灰度、8 位、24 位 RGB 等。TIFF 最初是无损格式，但是新的 JPEG 标记能让你选择使用 JPEG 压缩。因为 TIFF 不像 JPEG 那样是用户可控的，所以它不具有后者主要优点。

### 3.2.5 EXIF

交换图像文件 (Exchange Image File, EXIF) 是用于数码相机的图像格式。它最初发明于 1995 年，当前的版本 (2.2) 在 2002 年由日本电子和信息技术工业联合会 (JEITA) 发布。压缩的 EXIF 文件采用基线 JPEG 格式。有多种标记 (比 TIFF 中的多很多) 可以用来支持高质量打印，因为有关照相机和照相环境 (闪光、曝光、光源、白平衡、场景类型等) 的信息能够存储下来并且供打印机用于可能的色彩校正算法中。EXIF 标准也包含用于数字图像伴随的声音的文件格式的规范。它也支持用于转换为 FlashPix (最初由 Kodak 开发) 所需信息的标记。

### 3.2.6 图形动画文件

一些主流格式的目标都是以存储图形动画 (如一系列的画图或者图形插图) 来对抗视频 (比如，一系列的图像)。二者区别在于，与视频文件相比，动画对于资源的要求相当少。然而，动画文件格式可以用来存储视频信息并且有时就是这么用的。

FLC 是一种重要的动画或者移动图片文件格式，它最初由 Animation Pro 创建。另一种格式 FLI 与 FLC 相似。

GL 能产生出质量更好的移动图片。GL 动画也能够处理大文件。

许多老一点的格式也能用于动画，像 DL、Amiga IFF，还可以应用 Apple Quicktime。当然，用于动画的 GIF89 文件也可以用于动画。

### 3.2.7 PS 和 PDF

PostScript 是一种用于排版的重要语言，许多高端的打印机有内置的 PostScript 解释器。PostScript 是基于向量的 (而不是基于像素的) 图片语言，其页元素本质上用向量定义。有了以这

种方式定义的字体, PostScript 包含文本, 也包含向量/结构化图形; 位映射图像也可以包含在输出图像中。封装的 PostScript 文件加上一些信息就可以将 PostScript 文件包含在另一个文档中。

一些流行的图形程序, 如 Illustrator 和 FreeHand, 都会用到 PostScript。然而, PostScript 页描述语言本身并没有提供压缩。实际上, PostScript 文件就是用 ASCII 保存的。因此, 文件通常都很大, 从理论上说, 这种文件通常只有在利用 UNIX 的工具如 compress 或者是 gzip 压缩之后才能够使用。

因此, 另一种文本+图的语言已经开始取代 PostScript, Adobe Systems 公司在它的便携式文档格式 (Portable Document Format, PDF) 中包含了 LZW (见第 7 章) 压缩。其结果是, 不包含图像的 PDF 文件拥有几乎相同的压缩比, 2:1 或者 3:1, 就像用其他基于 LZW 压缩工具的文件一样, 比如 UNIX 的 compress 或是 gzip 基于 PC 的 winzip (pkzip 的一个变种)。对于包含图像的文件, PDF 通过对图像内容使用单独的 JPEG 压缩 (和创建原始图和压缩版本的工具有关), 可以获得更高的压缩比。Adobe Acrobat PDF 阅读器也可以用于阅读构建为超链接元素的文档, 这种文档提供可点击的内容和方便的树状结构的总结的链接图表。

### 3.2.8 Windows WMF

Windows MetaFile (WMF, Windows 元文件) 是用于 Microsoft Windows 操作环境的本地向量文件格式。WMF 文件实际上由一系列图形设备接口 (Graphics Device Interface, GDI) 函数调用组成, 它对 Windows 环境而言也是本地的。当一个 WMF 文件被“播放” (通常使用 Windows 的 PlayMetaFile() 函数), 所描述的图形便被呈现。WMF 文件表面上是设备无关和大小无限制的。

### 3.2.9 Windows BMP

位图 (BitMap, BMP) 是 Microsoft Windows 主要的系统标准图形文件格式, 用在 Microsoft Paint 和其他程序中。它使用游长编码压缩 (见第 7 章), 并能够有效地存储 24 位位图图像。然而, 要注意的是, BMP 包含了许多方式, 包括非压缩的 24 位图像。

### 3.2.10 Macintosh PAINT 和 PICT

PAINT 最初是在 MacPaint 程序中使用, 一开始只支持 1 位单色图像。

PICT 在 MacDraw (基于向量的画图程序) 中用于存储结构化图形。

78

### 3.2.11 X Windows PPM

这是用于 X Windows 系统的图形格式。可移植的像素映射 (Portable PixMap, PPM) 支持 24 位彩色位图, 并能够用许多公共领域的图形编辑器操作, 比如 xv。它在 X Windows 系统中用于存储图标、像素映射、背景等。

## 3.3 进一步探索

Foley 等[2]对计算机图形学提供了详尽的介绍。图像处理问题的更好的讨论见 Gonzalez 和 Woods 的著作[7]。包含当前常用文件格式的列表的更多信息可以在本书的网页上找到, 在第 3 章“Further Exploration”目录下可以找到这些信息。

其他的链接包括:

- GIF87 和 GIF89 细节。虽然这些文件格式不是那么有趣, 但它们很简单, 而且可以从中学位流怎样开始的。
- 用于开发 GIF 动画的流行共享软件。

- JPEG 详细考虑。
- PNG 细节。
- PDF 文件格式。
- 普适的 BMP 文件格式。

根据这些文件格式实际的输入/输出，网站上有用于读和操作简单的 24 位 BMP 文件的代码。

### 3.4 练习

1. 简要解释为什么我们需要少于 24 位的颜色，而且这样为什么会带来麻烦？一般而言，我们需要怎样做才能把 24 位颜色值转换为 8 位颜色值？
2. 假设我们需要量化 8 位的灰度图像到仅仅 2 位精度。最简单的办法是什么？原图像中字节值的哪个范围映射到哪个量化值？
3. 假设我们有 5 位灰度图像。我们需要多大的抖动矩阵来在 1 位打印机上显示这幅图？
4. 假设对一幅彩色图像的每一个像素，我们有 24 位可以利用。然而，我们发现，比起蓝色来，人们对红色和绿色更敏感，实际上，人们对红色和绿色的敏感度是蓝色的 1.5 倍。我们怎样才能最好地使用可以利用的位？
5. 在你的工作中，你决定通过用更多的磁盘空间来存储公司的灰度图像，以给你的老板留下深刻印象。你喜欢用 RGB，每个像素 48 位，而不是每个像素使用 8 位。那么，你怎样用新的格式存储原来的灰度图像以使它们在视觉上看起来和原来一样呢？
6. 有时，一幅图像的位平面可以用地图绘制中“海拔”这个术语来刻画。图 3-18 显示了一些海拔值。

假设我们用 8 位平面来描述 8 位图像。简短讨论你怎样才能根据地理概念来观看每个位平面。

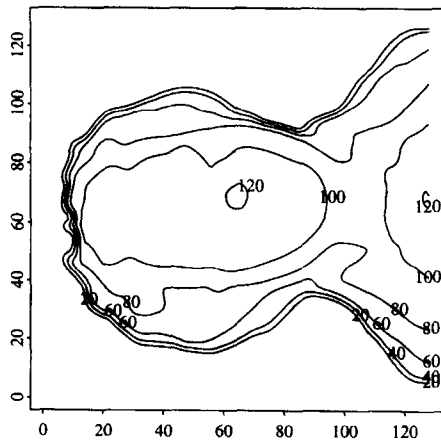


图 3-18 地理上的海拔

7. 对于颜色查找表问题，在一幅图像上试试中值区分算法。简要解释该算法若用在一幅红苹果图像上，在红颜色中，为什么把更多的颜色等级放在那些需要 24 位彩色图像的地方？
8. 关于无序抖动，一本标准的图形教材[2]讲到：“甚至可以使用更大的模式，但是空间分辨率与亮度分辨率的折中受到我们视觉敏锐程度（普通光中大约 1' 的弧）的限制。”
  - (a) 这句话是什么意思？
  - (b) 如果拿着一张纸站在一英尺开外的地方，点之间的大致线性距离是多少？（提示：1'

的弧是 $1^\circ$ 的角的 $1/60$ 。一个圆上的弧长等于角（用弧度）乘以半径。）我们能够看到用300dpi打印机打出的点的间隙吗？

80

(c) 写一个算法（伪代码）来计算 RGB 数据的颜色直方图。

### 3.5 参考文献

- [1] J. Miano, *Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP*, Reading, MA: Addison-Wesley, 1999.
- [2] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice in C*, 2nd ed., Reading, MA: Addison-Wesley, 1996.
- [3] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, Boston: PWS Publishing, 1999.
- [4] L.G. Shapiro and G.C. Stockman, *Computer Vision*, Upper Saddle River, NJ: Prentice Hall, 2001.
- [5] P. Heckbert, "Color Image Quantization for Frame Buffer Display," in *SIGGRAPH Proceedings*, vol. 16, p. 297-307, 1982.
- [6] W.B. Pennebaker and J.L. Mitchell, *The JPEG Still Image Data Compression Standard*, New York: Van Nostrand Reinhold, 1993.
- [7] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2002.

81

## 第4章 图像和视频中的颜色

彩色图像和视频在网络和多媒体产品中随处可见。人们日益关注见到的颜色与屏幕上所显示的颜色之间存在差异的问题。HTML 标准最新的版本试图解决这个问题，它按照一个由色彩方面的科学家制定的标准“sRGB”来指定颜色。

为了了解简单而又奇妙的色彩世界，我们在本章中将介绍如下主题：

- 颜色科学。
- 图像中的颜色模型。
- 视频中的颜色模型。

### 4.1 颜色科学

#### 4.1.1 光和光谱

回忆高中时学过的内容，光是一种电磁波，它的颜色由波的波长决定。激光由单波长组成，比如，红宝石激光能发出明亮的鲜红的光束。所以，如果我们要将光的亮度与波长之间的关系用图表示，我们将看到的是在适当的红色波长处的一个尖峰而没有其他波长的贡献。

相比之下，大多数光源在许多波长都有贡献。人类不可能看到所有的光，仅能看到对可见波长有贡献的部分。短波长产生蓝色感觉，而长波长产生红色感觉。

我们用一个叫分光光度计（spectrophotometer）的仪器来测量可见光（visible light），该仪器反射从一个衍射光栅（表面有刻度）发出的光。衍射光栅使不同波长的光分开，就像棱镜一样。图 4-1 展示了一个现象：白光含有彩虹所有的颜色。如果你透过棱镜看，你会注意到它会产生彩虹的效果，这是由于自然界的散射（dispersion）现象造成的。在肥皂泡表面能看到相似的现象。

可见光是波长在 400~700nm 的电磁波（这里 nm 表示纳米，即 $10^{-9}$ 米）。图 4-2 显示了在晴朗的天气典型的户外光在每个波长间隔的相对能量。这种被称为光谱能量分布（SPD）或者是光谱（spectrum）的曲线显示了每个波长的光的能量（电磁信号）的相对数量。波长符号是 $\lambda$ ，所以这种曲线可以表示为  $E(\lambda)$ 。

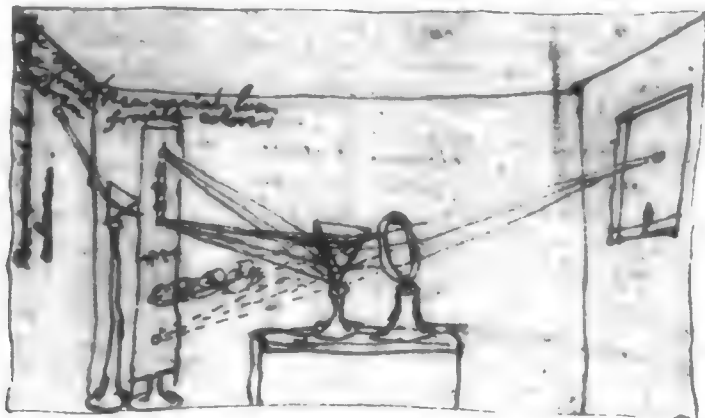


图 4-1 Isaac Newton 先生的实验（经牛津大学 New College 的 Warden 和 Fellows 的同意）



在实践中，通常用以下方法测量：把一个小的波长范围（如 5~10nm）内的电压数加起来，所以这样的图通常由每 10nm 一个的函数值连接起来的段组成。这也意味着这样的轮廓可以用向量存储。尽管在实际中，积分用求和来计算，但在下面的方程中，我们把  $E(\lambda)$  看作连续函数。

#### 4.1.2 人的视觉

眼睛就像一个照相机一样，晶状体把图像聚焦在视网膜上（但此时图像的上下和左右都是颠倒的）。视网膜由一组柱状细胞和三种视锥细胞组成，它们都是因其形状而得名。当光线亮度较低的时候柱状细胞发挥作用，制造出灰度阴影图像（“在晚上，所有的猫都是灰的”）。在光线亮度高的时候，每个视锥细胞都产生一个信号。因为它们的色素不同，所以三种视锥细胞对红（R）、绿（G）和蓝（B）光线最敏感。

更高的光线级层将使更多的神经元激发起来，但是按照这个思路思考，在大脑中究竟发生了什么却是一个仍在争论的课题。然而，大脑似乎利用了  $R-G$ 、 $G-B$ 、 $B-R$  之间的差异，当然也有将  $R$ 、 $G$  和  $B$  结合进高的光线级层的消色差通道（这样，我们就能说大脑的代数学不错）。

#### 4.1.3 眼睛的光谱灵敏度

眼睛对可见光谱中间部分的光最为敏感。就像一个光源的光谱能量分布轮廓，如图 4-2 所示，对接收器，我们显示了相对灵敏度作为波长的函数。蓝色接收器的灵敏度没有按照比例来显示，因为它比红色和绿色的曲线要小很多。蓝色在演化曲线中是一个持续很晚的添加。（而且，据统计，不管哪个国家，蓝色是人类钟爱的颜色——也许就是这个原因吧。蓝色真是有点让人惊奇！）图 4-3 用虚线显示了灵敏度的总和，该虚线称为发光效率函数。它通常用  $V(\lambda)$  表示，是红色、绿色和蓝色响应曲线的总和[1 和 2]。

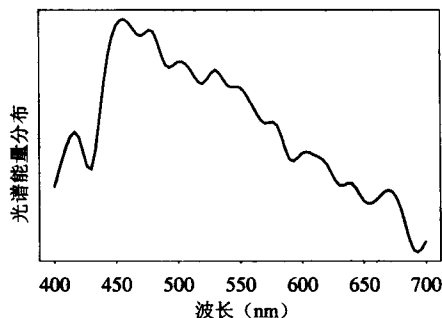


图 4-2 日光的光谱能量分布

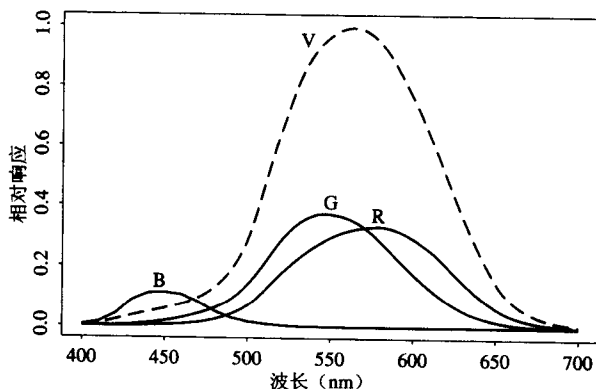


图 4-3 视锥细胞灵敏度：R、G 和 B 视锥细胞，以及发光效率曲线  $V(\lambda)$

柱状细胞对大范围的波长敏感，但是产生的信号只对黑白敏感。柱状细胞的灵敏度曲线看起来和发光效率函数  $V(\lambda)$  类似，但是向光谱的红端略有偏移。

眼睛有大约 600 万个视锥细胞，但是  $R$ 、 $G$  和  $B$  视锥细胞所占的比例是不同的。它们的比例大约为 40:20:1（参阅[3]可得到详细的解释）。所以由视锥细胞产生的消色差通道类似  $2R+G+B/20$ 。

光谱灵敏度函数通常用非  $R$ 、 $G$ 、 $B$  的字母表示, 所以这里让我们用向量函数  $q(\lambda)$  来表示它们, 各分量为

$$q(\lambda) = [q_R(\lambda), q_G(\lambda), q_B(\lambda)]^T \quad (4.1)$$

也就是说, 这里有三个传感器 (因此向量下标  $k=1,2,3$  适用), 每个传感器都是波长的函数。

眼睛里每个颜色通道的反应与神经元激发的数量成比例。对红色通道来说, 落入图 4-3 所示的红色视锥函数的非零区域的光线都将产生反应。所以红色通道的总反应是落入视网膜的红色视锥细胞敏感的光线的和, 权值由那里波长的灵敏度决定。再一次把这些灵敏度看作连续函数, 我们就能用积分的形式把这个思想简明地表达出来:

$$\begin{aligned} R &= \int E(\lambda) q_R(\lambda) d\lambda \\ G &= \int E(\lambda) q_G(\lambda) d\lambda \\ B &= \int E(\lambda) q_B(\lambda) d\lambda \end{aligned} \quad (4.2)$$

因为被传递的信号由三个数组成, 颜色形成了三维向量空间。

#### 4.1.4 图像的形成

上面的式 (4.2) 实际上只有在我们看一个自己发光的物体 (比如, 一盏灯) 时才成立。在大多数情况下, 我们假设光是从物体表面反射得到的。物体表面在不同的波长处所反射的光的能量是不同的, 深色表面反射的能量少于浅色表面。图 4-4 展示了从橙色运动鞋和褪色的蓝色牛仔褲的表面反射的光谱 [4]。反射函数用  $S(\lambda)$  表示。

成像的情形如下所示: 具有光谱能量分布  $E(\lambda)$  的光源发出的光照射到一个具有表面光谱反射函数  $S(\lambda)$  的表面后被反射, 然后被眼睛的视锥函数  $q(\lambda)$  过滤。基本的过程如图 4-5 所示。函数  $C(\lambda)$  称为颜色信号 (color signal), 是光源  $E(\lambda)$  与反射  $S(\lambda)$  的乘积:  $C(\lambda) = E(\lambda)S(\lambda)$ 。

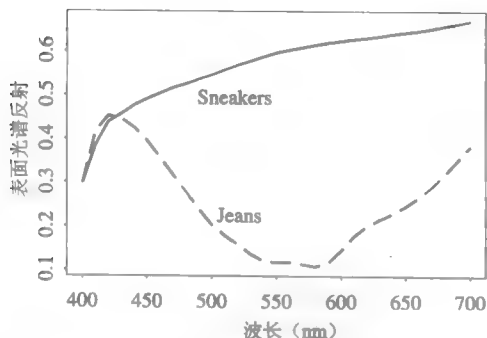


图 4-4 两个物体的表面光谱反射函数  $S(\lambda)$

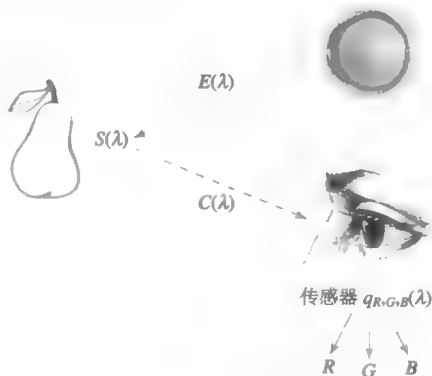


图 4-5 成像模型

类似于式 (4.2) 的考虑成像模型的方程为:

$$\begin{aligned} R &= \int E(\lambda) S(\lambda) q_R(\lambda) d\lambda \\ G &= \int E(\lambda) S(\lambda) q_G(\lambda) d\lambda \\ B &= \int E(\lambda) S(\lambda) q_B(\lambda) d\lambda \end{aligned} \quad (4.3)$$

### 4.1.5 照相机系统

现在,人类发明了以相似方式工作的照相机系统。好的照相机在每个像素位置(与视网膜的位置对应)产生三个信号。模拟信号转变为数字(截取为整数),并被存储起来。如果使用8位精度, $R$ 、 $G$ 、 $B$ 的最大值为255,最小值为0。

然而,进入电脑用户的眼睛的光是屏幕产生的——屏幕本质上是一个自发光的光源。因此,我们需要知道进入眼睛的光线 $E(\lambda)$ 。

85  
86

### 4.1.6 伽马校正

图像文件中的RGB数值被转换回模拟信号并驱动阴极射线管(Cathode Ray Tube, CRT)中的电子枪。电子的发射与驱动电压成比例,并且我们希望CRT系统产生的光线与电压线性相关。但情况并非如此。CRT产生的光线实际上与电压的指数大致成正比;这个指数称为“gamma”(伽马),符号为 $\gamma$ 。

如果文件在红色通道的值是 $R$ ,则屏幕发射的光线与 $R^\gamma$ 成正比,光谱能量分布(SPD)等于屏幕上(红色通道电子枪的目标)的红色荧光颜料的分布。 $\gamma$ 值大致在2.2左右。

因为电视接收器的工作机制与计算机的CRT相似,电视系统在传送电视电压信号之前通过实施反变换来预修正这种情况。通常是对需“伽马校正”的信号在发射前将其指数变为 $1/\gamma$ 。这样,我们就有:

$$R \rightarrow R' = R^{1/\gamma} \Rightarrow (R')^\gamma \rightarrow R \quad (4.4)$$

因此可以获得“线性信号”。

电压通常规范化到最大值为1,来看看伽马变换对信号的影响。图4-6a显示了没有应用伽马校正的输出光。我们可以看到,深一点的值的显示就太深了。图4-7a也显示了这种情况,该图显示了一个从左到右的线性坡道。

图4-6b显示了通过应用指数定律 $R^{1/\gamma}$ 的预校正信号的效果,通常规范化电压在0~1区间。我们看到,在4-6b中首先实施了校正,接着是4-6a中CRT系统的影响,如此则产生了线性信号。图4-7b显示了综合效果。这里,斜坡用16步表示,范围从灰度级0到灰度级255。

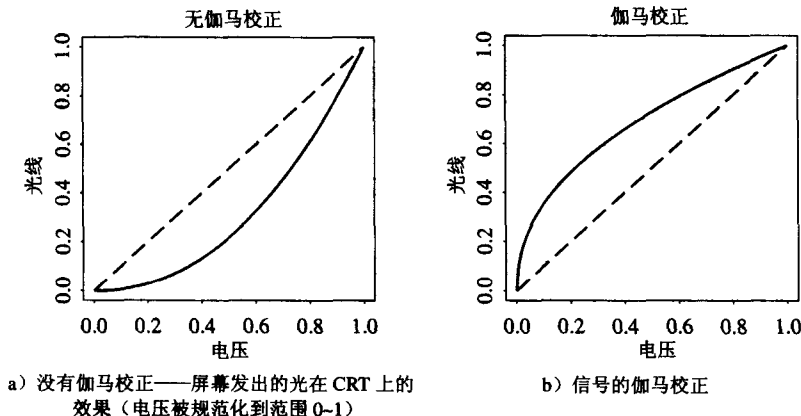
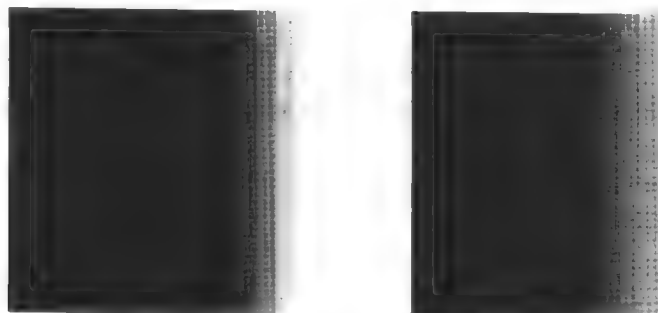


图4-6  $\gamma$ 校正的效果

87

更精确的 $\gamma$ 定义发现:简单的指数定律会在0电压处导致无穷的导数,这会造成很难设计一个完成伽马校正的模拟线路。实践中,采用更通用的变换,比如 $R \rightarrow R' = a \times R^{1/\gamma} + b$ ,在原点处有特别处理:



a) 从 0~255 的斜坡显示, 没有伽马校正

b) 使用了伽马校正的图像

图 4-7 伽马校正的效果

$$V_{\text{out}} = \begin{cases} 4.5 \times V_{\text{in}} & (V_{\text{in}} < 0.018) \\ 1.099 \times (V_{\text{in}} - 0.099) & (V_{\text{in}} \geq 0.018) \end{cases} \quad (4.5)$$

这称为摄像机变换函数 (camera transfer function), 上面这个定律被美国电影电视工程师学会 (Motion Picture and Television Engineers, SMPTE) 推荐为 SMPTE-170M 标准。

为什么  $\gamma$  值是 2.2 呢? 实际上, 这个值并不能产生最终指数为 1.0 的定律。这个数的历史被掩藏在电视机发明时美国国家电视系统委员会 (National Television System Committee, NTSC) 的决定之中。对彩色接收器, 指数定律实际上更接近 2.8。然而, 如果我们对这个指数定律仅仅补偿 2.2, 我们得到 1.25 的整体结果而不是 1.0。想法是: 在充满昏暗的观察条件下, 这样一个整体的转换产生出更令人满意的图像, 虽然带有颜色错误——深的颜色变得更深, 眼-脑系统也改变了对浅色和深色的相对对比度[5]。

随着基于 CRT 的计算机系统的出现, 情况就变得更加有趣。摄像机可能插入 (也可能没有插入) 伽马校正; 软件可能用某个伽马值来写入图像文件; 软件可能对期望的某个 (其他) 伽马解码; 图像存储在帧缓存中, 通常在帧缓存中为伽马提供一个查找表。毕竟, 如果我们用计算机图形学来产生图像, 没有使用伽马, 但是仍然需要用伽马对显示进行预补偿。

那么, 定义一个考虑了所有这些变换的全局“系统”伽马是有意义的。遗憾的是, 我们必须经常对全局伽马值进行猜测。Adobe Photoshop 允许我们尝试不同的伽马值。在进行 WWW 发布的时候, 知道 Macintosh 在它的图形卡里面进行了伽马值为 1.8 的伽马校正是很重要的。SGI 机希望的伽马是 1.4, 而大多数 PC 机或者是 Sun 工作站没有做额外的伽马校正, 并且可能显示伽马值为 2.5。因此, 对于最常见的机器, 用 Macintosh 和 PC 机的平均值 (大约 2.1) 来对图像进行校正是有意义的。

然而, 大多数从业者使用的值可能是 2.4, 这个值也被标准 RGB (sRGB) 组织所采纳。WWW 应用的一个新的“标准”RGB 称为 sRGB, 包含在将来所有的超文本标记语言 (HTML) 标准中。它定义了典型光线级别的标准模型和显示器条件, (或多或少) 是用于 Internet 的设备无关的颜色空间。

与伽马校正相关的一个问题是: 如何决定在一个文件的像素值中, 什么样的亮度级由什么样的位模式来表现。眼睛对亮度级比率, 而不是绝对亮度最敏感。这意味着光线越亮, 亮度级的改变也必须越大, 才能够感知到这种改变。

如果我们能精确控制表现亮度的位, 那么用算法对亮度进行编码以使可利用的位能最大限度地使用是很有意义的。那么, 我们应该包括使用  $1/\gamma$  指数定律变换的那个编码, 如式 (4.4) 所示,

或者执行这样的逆函数（见[6]的 564 页）的查找表。

然而，最有可能的是，我们遇到的图像或者视频没有位级的非线性编码，而是由便携式摄像机产生，或是用于广播电视。这些图像将根据式（4.4）进行伽马校正。国际照明协会（CIE）发起的基于色差标准的 CIELAB 提供了用于包含人类亮度感觉的非线性方面的精细的算法，详细讨论见 4.1.14 节。

#### 4.1.7 颜色匹配函数

事实上，许多颜色应用程序涉及指定和重新创建符合特定要求的颜色。假设你希望在屏幕上复制一个特殊的阴影，或是染色布的特殊深浅。多年来，甚至在图 4-3 的眼睛-灵敏度曲线产生之前，一门包含于心理学中的技术把基本的  $R$ 、 $G$ 、 $B$  光线的组合与某种色调匹配起来。三种基本光线可以组成特殊集合，这个集合称为原色（color primaries）集。为了与给定的色调匹配，要求一组观察者利用控件分别调节三原色的亮度，直到最终的光线的点与要求颜色最为接近为止。图 4-8 展示了基本的情形。执行这样一个实验的装置称为色度计（colorimeter）。

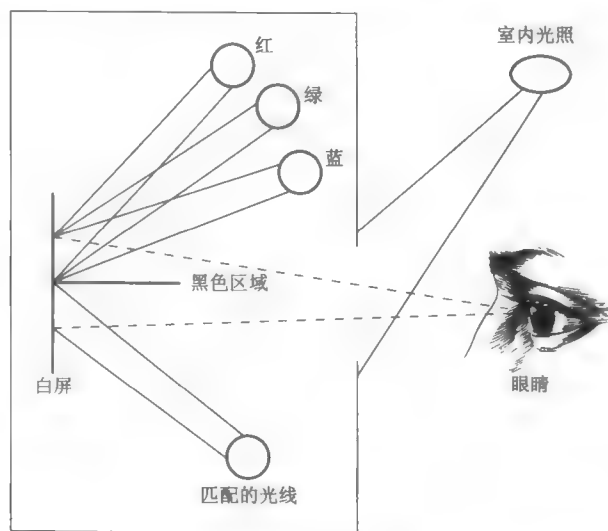


图 4-8 色度计实验

颜色的国际标准组织，国际照明组织（Commission Internationale de L'Eclairage, CIE），在 1931 年用一个称为颜色匹配函数（color-matching function）的曲线集中了所有这些数据。他们使用的原色在 440、545 和 580 纳米处具有峰值。假设你对匹配已知波长的激光（即单色的）而非布样感兴趣，那么颜色匹配实验由需要用在光的每个单独的窄带波长的原色的比例来描述，则一般光由单波长结果的线性组合匹配。

图 4-9 显示了 CIE 的颜色匹配曲线，用  $\bar{r}(\lambda)$ 、 $\bar{g}(\lambda)$ 、 $\bar{b}(\lambda)$  表示。实际上，这个曲线是远离图 4-3 眼睛灵敏度的线性矩阵乘积。

为什么曲线的某些部分是负的呢？这说明一些颜色不能由原色的线性组合来产生。对这样的颜色，一种或更多原色的光线必须从图 4-8 黑色部分的一端移到另一端，那么它们照亮了被匹配的样例而不是白屏。这样，从某种意义上说，这种样例被负光匹配。

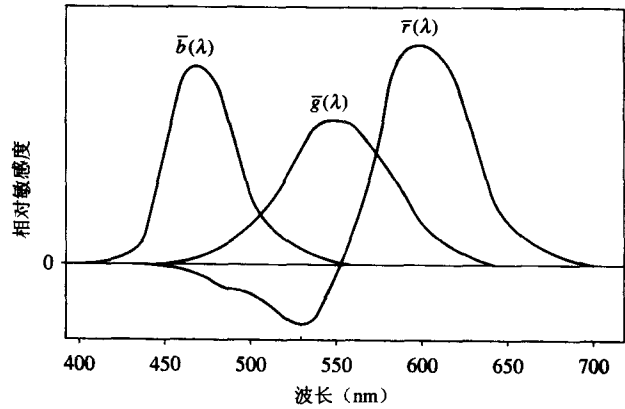


图 4-9 CIE 颜色匹配函数  $\bar{r}(\lambda)$ ,  $\bar{g}(\lambda)$ ,  $\bar{b}(\lambda)$

4.1.8 CIE 色度图

以前，工程师们发现，图 4-9 所示的 CIE 颜色匹配曲线有负的圆形突出部分，这让人感到不快。因此，设计了一套虚拟原色以使颜色匹配函数只有正值。图 4-10 显示了结果曲线，它们通常被称为颜色匹配函数。它们是从  $\bar{r}$ 、 $\bar{g}$ 、 $\bar{b}$  曲线进行线性 ( $3 \times 3$  矩阵) 变换而得到的，用符号  $\bar{x}(\lambda)$ 、 $\bar{y}(\lambda)$ 、 $\bar{z}(\lambda)$  表示。变换矩阵是根据以下条件选择的：中间的标准颜色匹配函数  $\bar{y}(\lambda)$  与图 4-3 所示的发光效率曲线  $V(\lambda)$  完全相等。

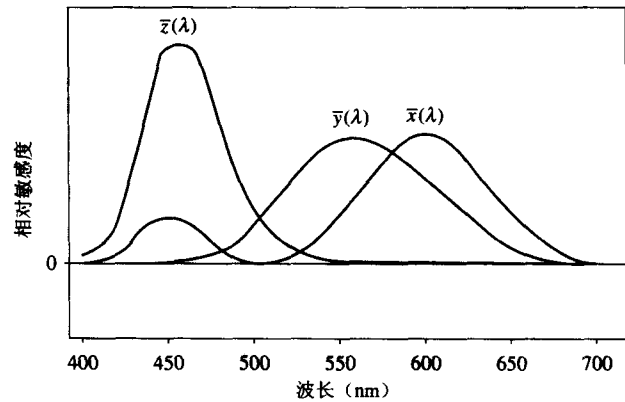


图 4-10 CIE 标准颜色匹配函数  $\bar{x}(\lambda)$ 、 $\bar{y}(\lambda)$ 、 $\bar{z}(\lambda)$

对于一个光谱能量分布  $E(\lambda)$  来说，表现颜色特征所需的基本色度信息是三色值 (tristimulus value)  $X$ 、 $Y$ 、 $Z$  的集合，其定义与式 (4.1) 类似，如下所示：

$$\begin{aligned} X &= \int E(\lambda) \bar{x}(\lambda) d\lambda \\ Y &= \int E(\lambda) \bar{y}(\lambda) d\lambda \\ Z &= \int E(\lambda) \bar{z}(\lambda) d\lambda \end{aligned} \tag{4.6}$$

中间值  $Y$  称为亮度 (luminance)。所有的颜色信息和变换都与这几个特殊值有关，它们包含了关于人类视觉系统的丰富信息。然而，显示 3 维数据十分困难，因此，CIE 设计了基于图 4-10

的曲线所暗含的  $(X, Y, Z)$  三元组的值的二维图。对每个可见的波长，这三条曲线值给出的  $X$ 、 $Y$ 、 $Z$  的值组成了人类所能看到的部分。然而，从式 (4.6) 我们观察到，增加亮度（增大灯泡的瓦特数）会为三色值增加了一个标量倍数。因此，通过某种办法得到向量  $(X, Y, Z)$  的量级来设计一个二维图是有意义的。在 CIE 系统中，这通过除以  $X+Y+Z$  来实现：

$$\begin{aligned}x &= X/(X+Y+Z) \\y &= Y/(X+Y+Z) \\z &= Z/(X+Y+Z)\end{aligned}\quad (4.7)$$

这充分地说明  $(x, y, z)$  外的其他值是多余的，因为我们有

$$x+y+z = \frac{X+Y+Z}{X+Y+Z} \equiv 1 \quad (4.8)$$

所以，

$$z = 1 - x - y \quad (4.9)$$

值  $x, y$  称为色度 (chromaticity)。

我们将每个三色向量  $(X, Y, Z)$  投影到点  $(1, 0, 0)$ 、 $(0, 1, 0)$  和  $(0, 0, 1)$  形成的平面上。通常，这个平面被视为投影到  $z=0$  平面，作为顶点  $(x, y)$  值为  $(0, 0)$ 、 $(1, 0)$ 、 $(0, 1)$  的三角形的内部点的集合。

图 4-11 显示了单色光的点的轨迹，画在这个 CIE “色度图” 上。沿着 “马脚” 底部的直线与可见光谱 (400 和 700nm, 从蓝色经绿色到红色) 极限处的点相交。这条直线称作紫色线 (line of purple)。“马脚” 本身称为光谱轨迹 (spectrum locus)，它显示了在每个可见波长处的单色光的  $(x, y)$  色度值。

颜色匹配曲线的作用是把相同的值 (每条曲线下的区域对每个  $\bar{x}(\lambda)$ 、 $\bar{y}(\lambda)$ 、 $\bar{z}(\lambda)$  是相同的) 加起来。因此对于一个所有光谱能量分布值都为 1 的白色光源 (“等能量白光”), 其色度值是  $(1/3, 1/3)$ 。图 4-11 显示了图中间的白点。最后，因为我们必须有  $x, y \leq 1$  和  $x+y \leq 1$ ，所以所有可能的色度值都必须位于图 4-11 的虚对角线之下。

注意，可以选择不同的 “白色” 光谱作为标准光源。CIE 定义了这样一些光源，比如光源 A、光源 C 和标准日光 D65 和 D100。它们都可显示为 CIE 图上略有不同的白点，D65 的色度等于  $(0.312713, 0.329016)$ ，而光源 C 具有色度值  $(0.310063, 0.316158)$ 。图 4-12 显示了这些标准光源的光谱能量分布曲线。光源 A 具有白炽光的特征，有钨丝灯的典型光谱能量分布，并且非常红。光源 C 是早期的表现日光的尝试，而 D65 和 D100 分别是中间范围和带蓝色的通常使用的日光。图 4-12 也显示了标准荧光光源尖钉似的光谱能量分布，这种光源被称为 F2[2]。

在光谱轨迹上具有色度的颜色表现出 “纯色”。它们是最 “饱和” 的，想一想蘸有墨水的纸变得越来越饱和。相比而言，靠近白点的颜色更加不饱和。

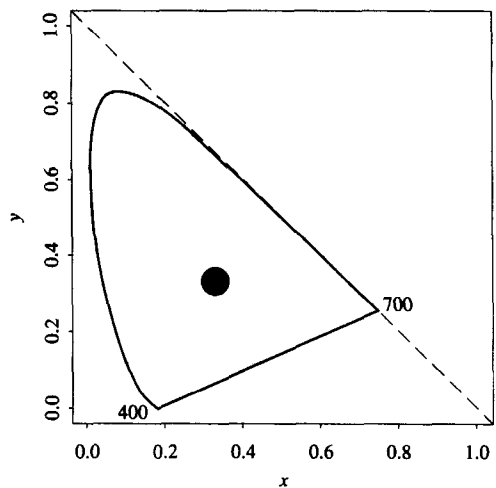


图 4-11 CIE 色度图

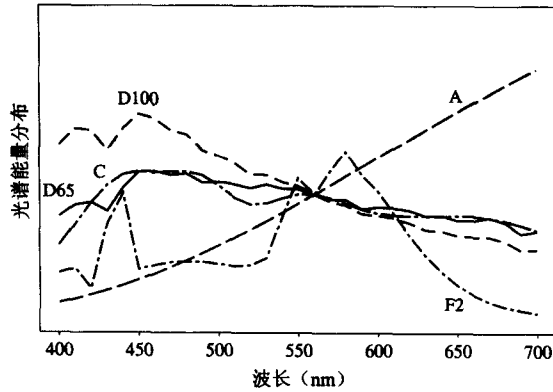


图 4-12 标准光源的光谱能量分布

对于两种光的混合，结果色度位于两种光的色度的连线上，这是色度图一个非常有用的性质。为严谨起见，我们没有说对所有颜色都是这样，仅仅是对光有上述结论。原因是，我们至今还坚持颜色混合的加法（additive）模型。这个模型对于光效果很好，也适用于另一种特殊的情况，即显示器颜色。然而，下面我们将看到，上述结论对于打印机颜色（见 4.2.5 节）不成立。

对于 CIE 图中的任何色度，主波长是光谱轨迹与连接白点和指定颜色的直线延长线的交点。（对于与紫色线相交的颜色，一个主波长的补为反向延长该线通过白点。）另一个有用的定义是给定制色的补色是通过白点的线上的所有颜色。最后，色纯度（excitation purity）是从白点到给定颜色的距离与主波长的比值，用百分数表示。

4.1.9 彩色显示器规范

如果 RGB 电子枪都在其最高能量状态被激活，彩色显示器在一定程度上由要求的白点色度确定。事实上，我们可能使用伽马校正值  $R'$ 、 $G'$ 、 $B'$ 。如果我们把电压规范化到 0~1，那么我们希望显示器能在  $R'=G'=B'=1$  时显示要求的白点（省略了从文件中的值到电压值的转换，仅仅陈述了像素颜色值规范化到最大值 1）。

然而，显示器屏幕内的荧光涂料具有自己的色度，所以乍一看，我们似乎不能控制显示器的白点。然而，可以通过设定每个电子枪的增益控制，使它们达到出现白点所要求的最大电压来修正这个问题。

目前，已经有一些显示器规范。显示器规范由固定的、制造商指定的显示器荧光层的色度和所需的标准白点组成。表 4-1 显示了三个常用规范中的相应值。NTSC 是标准的北美和日本的规范。SMPTE 是它的更高版本，其中光源由光源 C 改为光源 D65，而荧光层色度与现代机器更加吻合。数字视频使用的规范与北美的规范类似。EBU 系统从欧洲广播联盟（European Broadcasting Union）派生而来，并用于逐行倒相（Phase Alternating Line, PAL）和顺序传送与存储彩色电视系统（System Electronique Couleur Avec Memoire, SECAM）视频系统中。

表 4-1 显示器规范的色度和白点

系统	红		绿		蓝		白 点	
	$x_r$	$y_r$	$x_g$	$y_g$	$x_b$	$y_b$	$x_w$	$y_w$
NTSC	0.67	0.33	0.21	0.71	0.14	0.08	0.3101	0.3162
SMPTE	0.630	0.340	0.310	0.595	0.155	0.070	0.3127	0.3291
EBU	0.64	0.33	0.29	0.60	0.15	0.06	0.3127	0.3291



#### 4.1.10 超色域的颜色

现在, 我们暂不考虑伽马校正, 那么显示颜色的一个重要问题是怎样产生与设备无关的 (device-independent) 颜色, 通过用与设备相关 (device-dependent) 的颜色值 RGB 来指定  $(x, y)$  色度值来协商解决。

对于每一对  $(x, y)$ , 我们都希望找到 RGB 三元组给出具体的  $(x, y, z)$ , 因此, 利用  $z=1-x-y$  得到光源的  $z$  值, 并且解决了制造商指定色度的 RGB。因为如果我们没有绿色或者蓝色值 (即文件值为 0), 那么我们只能看红色光源的色度, 我们通过下面公式组合成  $R$ 、 $G$ 、 $B$  的非零值:

$$\begin{bmatrix} x_r & x_g & x_b \\ y_r & y_g & y_b \\ z_r & z_g & z_b \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (4.10)$$

如果  $(x, y)$  是指定的而非从上式推导得到的, 我们就要把光源  $(x, y, z)$  值的矩阵转置, 来得到正确的 RGB 值, 以获得要求的色度。

但是, 如果 RGB 数值中的某个值为负会怎样呢? 这种问题在人们能够感知到颜色, 但这种颜色在所使用的设备上不能被表现的情况下出现。这时, 我们说该颜色是超色域 (out of gamut) 的, 因为所有能显示的颜色集合组成了该设备的色域范围。

处理这种情形的一个办法是仅使用最接近的色域内的可用颜色。另一种常用方法是选择最接近的补色。

对一个显示器, 每种可以显示的颜色都位于一个三角形中。这是从 Grassman 定律得出的, 它描述了人类的视觉, 说明“颜色匹配是线性的”。这意味着由三原色组成的光线的线性组合仅是用来使组合乘上那些原色的权值的线性集合。也就是说, 如果我们用从三个光源发出的三条光线的线性组合来组成颜色, 我们仅能从这些光线的凸集 (我们将在下面看到, 对于打印机, 这个凸面不再成立) ——这种情况下是一个三角形——来创造颜色。

图 4-13 显示了 CIE 图中的 NTSC 系统的三角形色域。假设这个小三角形表现一个要求的颜色, 则在 NTSC 显示器色域边界上的点是连接要求颜色与白点的线与形成三角形边界的最近的线的交点。

#### 4.1.11 白点校正

我们至今所做的工作中的一个不足之处是我们要把三色值 XYZ 映射到设备的 RGB, 而不仅是处理色度  $xyz$ 。区别在于 XYZ 值包含颜色的量级。我们也要在  $R$ 、 $G$ 、 $B$  都达到最大值时, 得到白点。

但是, 表 4-1 会产生错误的值。考虑 SMPTE 规范, 设  $R=G=B=1$ , 则  $X$  的值等于  $x$  值的和, 或者说是  $0.630+0.310+0.155=1.095$ 。相似的,  $Y$  与  $Z$  的值结果为 1.005 和 0.9。用  $(X+Y+Z)$  来除, 得到色度  $(0.365, 0.335)$  而非要求的值  $(0.3127, 0.3291)$ 。

要修正这两个缺陷, 要先把  $Y$  的白点量级设为 1:

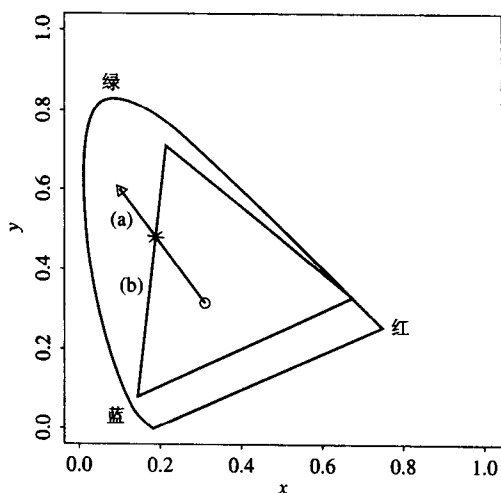


图 4-13 用色域内颜色近似色域外颜色。三角形显示的色域外颜色用从那个颜色到白点的线与设备色域的边界的交点近似

$$Y(\text{白点})=1 \quad (4.11)$$

现在我们需要找到三个修正因子的集合, 这样, 如果用三个电子枪所得到的增益乘上这些值, 就可以得到  $R=G=B=1$  处的白点 XYZ 值。假设式 (4.10) 中的光源色度  $x_r, x_g, \dots$  的矩阵称为  $M$ , 把修正表示为对称矩阵  $D=\text{diag}(d_1, d_2, d_3)$ , 得到

$$XYZ_{\text{white}} \equiv MD(1,1,1)^T \quad (4.12)$$

这里  $( )^T$  意为转置。

对于 SMPTE 规范, 我们有  $(x, y, z) = (0.3127, 0.3291, 0.3582)$ , 或者除以中间值,  $XYZ_{\text{white}} = (0.95045, 1, 1.08892)$ 。我们注意到  $D$  乘以  $(1, 1, 1)^T$  得到  $(d_1, d_2, d_3)^T$ , 我们最后用一个指定的  $(d_1, d_2, d_3)^T$ :

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{\text{white}} = \begin{bmatrix} 0.630 & 0.310 & 0.155 \\ 0.340 & 0.595 & 0.070 \\ 0.03 & 0.095 & 0.775 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \quad (4.13)$$

利用上面新得到的值  $XYZ_{\text{white}}$ , 并将结果转置, 得到

$$(d_1, d_2, d_3) = (0.6247, 1.1783, 1.2364) \quad (4.14)$$

#### 4.1.12 XYZ 到 RGB 的转换

现在, 从 XYZ 到 RGB 的  $3 \times 3$  的变换矩阵为

$$T=MD \quad (4.15)$$

对于不是白点的点, 也有:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = T \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (4.16)$$

对于 SMPTE 规范, 我们有:

$$T = \begin{bmatrix} 0.3935 & 0.3653 & 0.1916 \\ 0.2124 & 0.7011 & 0.0866 \\ 0.0187 & 0.1119 & 0.9582 \end{bmatrix} \quad (4.17)$$

把上式展开, 得到

$$\begin{aligned} X &= 0.3935 \cdot R + 0.3653 \cdot G + 0.1916 \cdot B \\ Y &= 0.2124 \cdot R + 0.7011 \cdot G + 0.0866 \cdot B \\ Z &= 0.0187 \cdot R + 0.1119 \cdot G + 0.9582 \cdot B \end{aligned} \quad (4.18)$$

#### 4.1.13 带伽马校正的转换

上面的计算假设我们处理的是线性信号。然而, 我们很有可能遇到非线性的伽马校正的  $R'$ 、 $G'$ 、 $B'$ , 而不是线性的  $R$ 、 $G$ 、 $B$ 。

进行 XYZ 到 RGB 变换的最好的办法是先计算转置式 (4.16) 所需要的线性 RGB 值, 然后通过伽马校正生成非线性信号。

然而, 情况通常并不是这样。相反, 对于  $Y$  值的等式是适用的但是用于非线性信号。实际上, 对于靠近白点的颜色, 这并不意味着有许多错误。对于精度的仅有让步是给从  $R'$ 、 $G'$ 、 $B'$  创造出

的新的  $Y$  值 一个新名字  $Y'$ 。  $Y'$  的重要性在于它对正在讨论的像素亮度的描述符进行了编码<sup>①</sup>。

最常用的变换方程是那些用于原始 NTSC 系统的基于光源 C 白点的方程，即使它们过时了。根据上面列出的方程，再结合表 4-1 中的值，我们得到下面的变换：

$$\begin{aligned} X &= 0.607 \cdot R + 0.174 \cdot G + 0.200 \cdot B \\ Y &= 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \\ Z &= 0.000 \cdot R + 0.066 \cdot G + 1.116 \cdot B \end{aligned} \quad (4.19)$$

这样，对非线性信号的编码从对非线性信号的亮度关联的编码开始：

$$Y' = 0.299 \cdot R' + 0.587 \cdot G' + 0.114 \cdot B' \quad (4.20)$$

(4.3 节更详细地讨论了非线性信号的编码。)

#### 4.1.14 L\*a\*b\*(CIELAB)颜色模型

上面关于怎样最好地使用我们可以利用的位的讨论涉及人类视觉对光线变化的感受的问题。从心理学角度来看，这个课题事实上是韦伯定律的一个例子，即数量越大，为了感知区别就必须有更多的变化。比如，当你抱起一个 4 岁的妹妹和一个 5 岁的弟弟，分辨他们的体重相对容易。然而，分辨两个重物就变得越来越困难。再看一个例子，要看出明亮光线中的变化，它们之间的变化必须比在昏暗光线中得到同样效果的变化大许多。这个现象的一个规则如下：感知到的同样变化必定是相对的。如果变化的比率是相同的，则感知到的变化是相同的，而不管是昏暗的光线还是明亮的光线。经过研究，从这个思想派生出一个算法雏形，用来感知相等的空间单位。

然而，对于人类视觉，CIE 得到了该规则的更复杂的版本，称为 CIELAB 空间。在该空间中被量化的仍然是感知到的颜色和亮度的不同。这是很有意义的，因为从实践上讲，颜色的区别对于比较源颜色和目标颜色很有用。举个例子，你可能对一批已染色的布是否与原始样品的颜色相同很感兴趣。图 4-14 显示了衡量颜色变化的坐标空间的 3 维立体剖面图。

CIELAB (也称为 L\*a\*b\*) 使用了  $1/3$  指数定律而不是一个算法。CIELAB 使用三个值，[它们大致与亮度以及组合起来制造色彩和色调的一对值对应 (带有星号的变量，以便与 CIE 以前的版本区分)。]色差的定义为：

$$\Delta E = \sqrt{(L^*)^2 + (a^*)^2 + (b^*)^2} \quad (4.21)$$

其中

$$\begin{aligned} L^* &= 116 \left( \frac{Y}{Y_n} \right)^{(1/3)} - 16 \\ a^* &= 500 \left[ \left( \frac{X}{X_n} \right)^{(1/3)} - \left( \frac{Y}{Y_n} \right)^{(1/3)} \right] \\ b^* &= 200 \left[ \left( \frac{Y}{Y_n} \right)^{(1/3)} - \left( \frac{Z}{Z_n} \right)^{(1/3)} \right] \end{aligned} \quad (4.22)$$

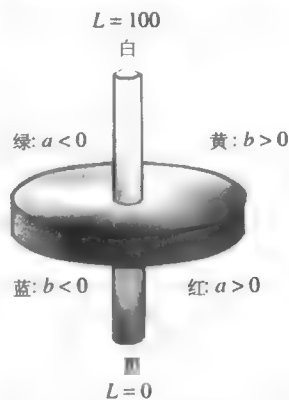


图 4-14 CIELAB 模型。(彩色插页中也有此图)

① 在文本网站的 Color FAQ 文件中，新值  $Y'$  被称为 “luma”。

$X_n$ 、 $Y_n$ 、 $Z_n$  是白点的 XYZ 值。相关定义为

$$\begin{aligned} \text{色度} = c^* &= \sqrt{(a^*)^2 + (b^*)^2} \\ \text{色调角} = h^* &= \arctan \frac{b^*}{a^*} \end{aligned} \quad (4.23)$$

大致地,  $a^*$  的最大值和最小值与红色和绿色一致; 而  $b^*$  变化范围是黄色到蓝色。色度 (chroma) 是色彩的规模, 在每个  $L^*$  亮度级, 更加多彩 (更饱和) 的颜色占据了 CIELAB 固体的外面部分, 而更褪色 (不饱和) 的颜色靠近中央的无色轴。色调角或多或少表达了大多数人说到“颜色”时的意思, 即, 你可以把它描述成红色或是橙色。

这种色差模型是研究的一个热门领域, 并且基于人类感知的公式过剩了 (另一个与 CIELAB 类似的竞争者称为 CIELUV——都是在 1976 年发明)。人们对这个领域产生兴趣, 一部分原因是由于这种颜色度量影响到我们怎样在不同的设备和网络界限上对光线与视觉的差别建模[7]。一些高端产品 (包括 Adobe Photoshop) 使用了 CIELAB 模型。

#### 4.1.15 其他颜色坐标系

还有其他一些坐标系统用于描述人类能感知的颜色, 但是在应该还是不应该使用伽马校正有一些混淆。这里, 我们描述的是与设备无关的颜色, 即基于 XYZ 并且和人类所见到的事物相关。然而, 人们通常把 RGB 和  $R'$ 、 $G'$ 、 $B'$  混用。

其他的颜色坐标系统包括 CMY (将在 4.2.2 节中介绍)、HSL (即色调、饱和度和亮度)、HSV (即色调、饱和度和值)、HSI (即色调、饱和度和亮度)、HCI (即色调、色度和亮度)、HVC (即色调、值和色度) 以及 HSD (即色调、饱和度和暗度) 等。

#### 4.1.16 蒙塞尔颜色命名系统

颜色的精确命名 (naming) 也是需要重视的一个问题。一个经受时间考验的标准系统由蒙塞尔 (Munsell) 在 20 世纪初设计, 并且进行过多次修改 (最后一个版本被称为 Munsell renotation) [8]。其思想是, 建立一个近似的感知规范系统, 它用三个坐标轴来讨论和指定颜色。这些坐标轴是值 (黑-白)、色调和色度。值被分成 9 阶, 色调围绕一个圆有 40 阶, 色度 (饱和度) 的最大阶数为 16。圆的半径随值而变化。

该系统希望对任何用户, 包括艺术家, 使用不变的颜色规范。蒙塞尔公司销售绘画所用的颜色书, 它们由关于绘画的公式组成 (这些书相当贵)。据称, 这是最常用的规范。

## 4.2 图像中的颜色模型

现在, 我们已经介绍了颜色科学, 以及与用于图像显示的颜色有关的一些问题。但是颜色模型和坐标系统怎样真正地用于存储、显示和打印图像呢?

### 4.2.1 CRT 显示器的颜色模型

从第 3 章可知, 我们通常以 RGB 形式存储颜色信息。然而, 前面的章节中介绍过, 这样一个坐标系统实际上是设备相关的。

对于足够精确的颜色, 我们希望每个颜色通道使用 8 位。实际上, 我们不得不为每个通道使用大约 12 位来避免深色图像区域的混淆现象——由伽马校正导致的轮廓结合, 因为伽马校正会导致可利用的整数级更少 (见练习 7)。

对于计算机图形产生的颜色, 我们把与亮度成比例的整数存储在帧缓存里面。然后, 在帧缓

存与 CRT 之间需要有一个伽马校正查找表。如果在存入帧缓存之前，伽马校正用在还没有被量化成整数的浮点数上，那么我们只能使用 8 位/通道，并且也能避免轮廓结合现象。

100

#### 4.2.2 减法法：CMY 颜色模型

到现在，我们已经仅有效地处理了加性颜色（additive color）。也就是说，当两条光线照射到一个目标上时，它们的颜色会加起来；当 CRT 屏幕上的两个光源被打开，它们的颜色也会相加。例如，红色光源+绿色光源会产生黄色光。

但是对于沉积到纸上的墨水，则会发生相反的情况，黄墨水从白色光源中减去蓝色，反射出红色和绿色，这就是为什么它看起来是黄色的原因！

所以，不用红色、绿色、蓝色这几种原色，我们需要一红色、一绿色和一蓝色的原色，即需要减去 R、G 或 B。这些减性原色是青（Cyan、C）、洋红（Magenta、M）和黄色（Yellow、Y）。图 4-15 显示了系统 RGB 和 CMY 是怎样联系的。在加性（RGB）系统中，黑色是没有光，RGB=(0, 0, 0)。在减性 CMY 系统中，黑色是使墨水的 C=M=Y=1 减去所有的光线产生的。

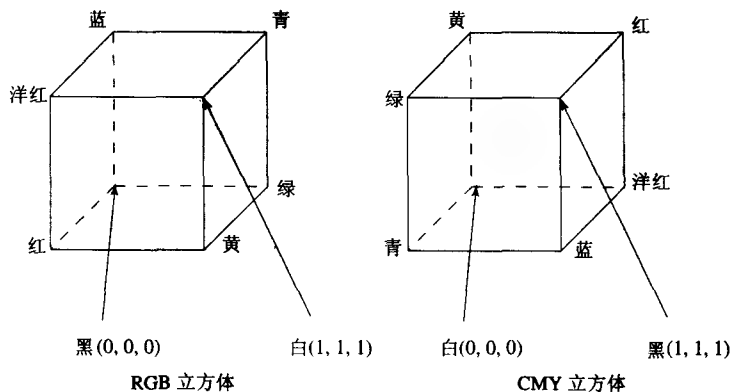


图 4-15 RGB 和 CMY 颜色立方体（彩色插页中也有此图）

#### 4.2.3 从 RGB 到 CMY 的转换

考虑墨水在减性系统中的作用，我们能创建的最简单模型是指定以怎样的密度把墨水放在纸上，来产生特定的 RGB 颜色，如下所示：

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (4.24) \quad 101$$

其逆变换是

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix} \quad (4.25)$$

#### 4.2.4 消除不足颜色：CMYK 系统

通常认为 C、M 和 Y 混合起来是黑色。然而，它们更常被混合成土褐色（我们都从幼儿园开始就知道这个结论）。真正“黑色”的黑墨水实际上比用混合彩色墨水来制作黑墨水便宜，所以

一个简单的产生准确的打印机颜色的方法是：计算三色混合中为黑色的部分，从颜色比例中去除之，用真正的黑色加回来。这被称为“消除不足颜色”。

因而新的墨水规范是

$$K = \min\{C, M, Y\}$$
$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} \Rightarrow \begin{bmatrix} C - K \\ M - K \\ Y - K \end{bmatrix}$$

(4.26)

图 4-16 描述了组合原色产生的颜色组合，有两种情况：加性颜色（通常使用 RGB 来组成颜色）和减性颜色（通常用 CMY 和 CMYK 来组成颜色）。

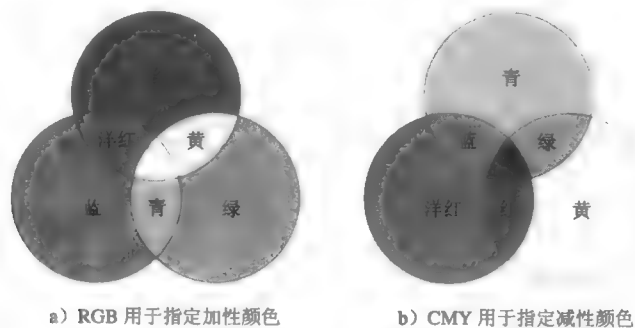


图 4-16 加性和减性颜色（彩色插页中也有此图）

4.2.5 打印机色域

在打印过程中，打印机把透明的墨水层放置到底层（通常是白色）上。如果我们希望青色墨水与缺红色的颜色真正相等，则我们的目标是产生能够完全吸收红色光但能完全通过绿色和蓝色光的墨水。遗憾的是，这种“块染色”仅是近似值。在实际中，传输曲线对 C、M 和 Y 墨水有交叠。这导致了颜色通道间的“色度亮度干扰”和预测打印时可利用颜色的困难。

图 4-17a 显示了块染色的典型传输曲线，图 4-17b 显示了使用这种墨水的彩色打印机的结果色域。我们看到，色域比 NTSC 显示器的小，并与之交叠。

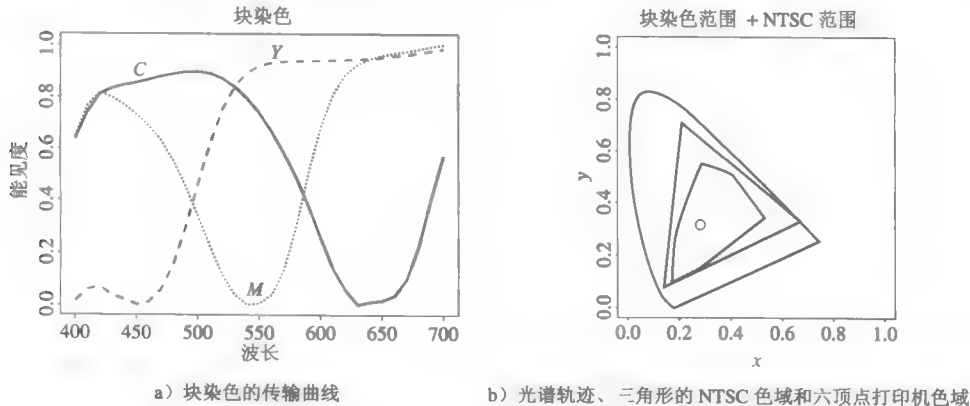


图 4-17

这种色域起源于用于打印机墨水的模型。传输与光学密度 (optical density) 相关, 即  $D = -\ln T$ , 这里  $T$  是图 4-17a 中的一条曲线。颜色是由墨水的  $D$  的线性组合构成的, 而  $D$  是三种权值分别为  $w_i (i=1,2,3)$  的密度的线性组合,  $w_i$  的范围从 0 到没有模糊的最大允许值之间。

因此, 总的传输  $T$  是三个带权密度的指数的乘积——光线通过透明染料的夹层时会有指数衰减。从纸面 (或是通过一小块胶片) 反射的光为  $TE = e^{-D} E$ , 这里  $E$  是照明光。用式 (4.6) 可生成图 4-17b 中的打印机色域。

打印机色域的中心是白-黑轴, 六个边界顶点对应 C、M、Y 和满密度的三个组合 CM、CY、MY。墨水密度较小的部分在图的中间。所有墨水满密度对应图中心的黑/白点, 该点用 “o” 标记。对于这些特殊的墨水, 该点具有色度  $(x, y) = (0.276, 0.308)$ 。

103

## 4.3 视频中的颜色模型

### 4.3.1 视频颜色转换

处理数字视频中颜色的方法大部分是从以前对电视颜色编码的方法中衍生来的。通常地, 亮度的一些版本与单个信号中的颜色信息相结合。比如, 与式 (4.19) 类似的矩阵变换方法 YIQ, 用来在北美和日本传输电视信号。这种方法也用于在这些国家的 VHS 视频磁带编码中, 因为视频磁带技术也使用 YIQ。

在欧洲, 视频磁带使用 PAL 或者 SECAM 编码, 它以使用 YUV 的矩阵变换的电视为基础。

最后, 数字视频大多使用 YCbCr 矩阵变换, 它于 YUV<sup>①</sup> 最接近。

### 4.3.2 YUV 颜色模型

最初, YUV 编码用于 PAL 模拟视频中。YUV 的一个版本现在用于数字视频的 CCIR<sup>②</sup> 601 标准中。

首先, 它对一个等于式 (4.20) 中的  $Y'$  的亮度信号 (对于伽马校正信号) 编码 (回想一下,  $Y'$  通常被称为 “luma”)。luma  $Y'$  与 CIE 的经伽马校正的亮度值  $Y$  相似, 但不完全相等。在多媒体中, 用户通常模糊这个区别, 简单地把两者都看成亮度。

同量级或亮度一样, 我们需要丰富的颜色标尺, 并且色度会涉及颜色和同一亮度的白色之间的差别。这可用色差  $U$ 、 $V$  来表现:

$$\begin{aligned} U &= B' - Y' \\ V &= R' - Y' \end{aligned} \quad (4.27)$$

从式 (4.20)、式 (4.27) 可得:

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ -0.299 & -0.587 & 0.886 \\ 0.701 & -0.587 & -0.144 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \quad (4.28)$$

我们通过转置式 (4.28) 的矩阵, 从  $(Y', U, V)$  得到  $(R', G', B')$ 。

注意, 对于  $R'=G'=B'$  的灰度像素, 亮度  $Y'$  等于相同灰度值的  $R'$ , 因为式 (4.20) 中的系数和为  $0.299+0.587+0.114=1.0$ 。对于灰度 (“黑和白”) 图像, 色度  $(U, V)$  是 0, 因为式 (4.28) 中后两行的系数和为 0。彩色电视能仅使用  $Y'$  信号<sup>③</sup> 在黑白电视上显示。为了向后兼容, 彩色电视通

① 亮度-色度颜色模型 (YIQ、YUV、YCbCr) 被证明是有效的。因此, 它们也被 JPEG 和 JPEG 2000 这样的图像压缩标准采纳。

② 需要注意的是, 许多作者和用户只使用  $Y$ , 没有加 “'” 号, 其义 (可能) 和  $Y'$  相同。

104 过指定带  $Y'$  的信号来使用不带颜色信息的的黑白信号。

最后, 在实际实现时, 为拥有更方便的最大值和最小值,  $U$  和  $V$  被重新调节。对模拟视频来说, 每个  $U$  或  $V$  都被限制在  $Y'[9]$  的最大值的  $\pm 0.5$  倍范围内。(注意, 真实电压是在另一个没有规范化的范围内——对模拟电压,  $Y'$  通常是在  $0 \sim 700\text{mV}$  的范围内, 所以重新调整后的  $U$  和  $V$  (在那个上下文中称为  $P_B$  和  $P_R$ ) 的范围为  $\pm 350\text{mV}$ 。)

这种缩放反映了怎样处理分量视频——三个分离的信号。然而, 对于处理复合 (composite) 视频, 即我们想用  $Y'$ ,  $U$  和  $V$  一次组合成单独的的信号, 在  $-1/3$  到  $+4/3$  范围内包含复合信号量级  $Y' \pm \sqrt{U^2 + V^2}$  是方便的, 这样它将保持在记录设备的幅度限制范围内。为达到这个目的,  $U$  和  $V$  重新调整为:

$$\begin{aligned} U &= 0.492111(B' - Y') \\ V &= 0.877283(R' - Y') \end{aligned} \quad (4.29)$$

(乘数有时四舍五入为三位有效数字), 则色度信号由  $U$  和  $V$  组成为复合信号:

$$C = U \cdot \cos(\omega t) + V \cdot \sin(\omega t) \quad (4.30)$$

这里  $\omega$  代表 NTSC 颜色频率。

从式 (4.29) 我们注意到, 0 并非  $U$  和  $V$  的最小值。根据真实的正的颜色, 在 RGB 立方体中,  $U$  大致从蓝 ( $U > 0$ ) 到黄 ( $U < 0$ ),  $V$  大致从红 ( $V > 0$ ) 到蓝绿 ( $V < 0$ )。

图 4-18 显示了一幅典型彩色图像分解为  $Y'$ 、 $U$  和  $V$  分量的结果。因为  $U$  和  $V$  都会为负, 所以这幅图实际上是真实信号经移动、重新调整后的版本。

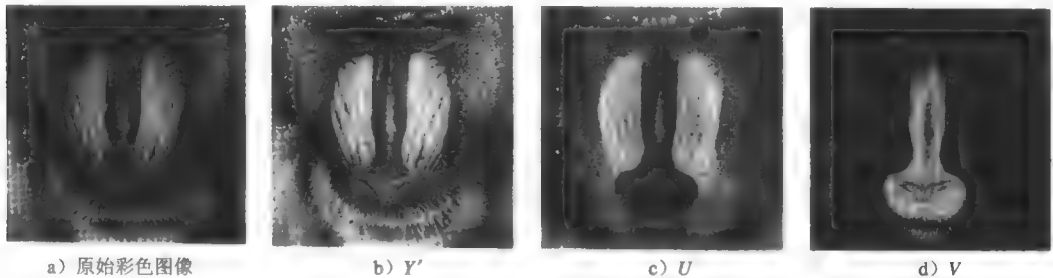


图 4-18 彩色图像的  $Y'UV$  分解 (彩色插页中也有此图)

因为眼睛对于黑白变换最为敏感, 对于空间频率 (例如, 与看细的彩色线条相比, 眼睛对细的灰度线条的格子看得更加清楚), 在模拟 PLA 信号中仅分配了  $1.3\text{MHz}$  的带宽给  $U$  和  $V$ , 而将  $5.5\text{MHz}$  的带宽留给  $Y'$  信号。实际上, 为彩色电视传输的颜色信号是非常斑驳的。

### 4.3.3 YIQ 颜色模型

YIQ (实际上是  $Y'IQ$ ) 用于 NTSC 彩色电视广播中。灰度像素产生零 ( $I$ ,  $Q$ ) 色度信号。这些名字的原始含义来源于模拟信号的组合,  $I$  表示同相色度,  $Q$  表示积分色度, 而现在可以忽略了。

我们认为, 虽然  $U$  和  $V$  的定义更简单, 但不能捕获到人类视觉灵敏度从最多到最少的层次。虽然它们很好地定义了色差, 但它们不能完全与人类感知的颜色灵敏度相符。NTSC 用  $I$  和  $Q$  来代替。

YIQ 是 YUV 的一个版本, 使用相同的  $Y'$ , 但是  $U$  和  $V$  旋转了  $33^\circ$ :

$$\begin{aligned} I &= 0.877283(R' - Y') \cos 33^\circ - 0.492111(B' - Y') \sin 33^\circ \\ Q &= 0.877283(R' - Y') \sin 33^\circ + 0.492111(B' - Y') \cos 33^\circ \end{aligned} \quad (4.31)$$

于是产生如下的矩阵变换:



$$\begin{bmatrix} Y' \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ 0.595879 & -0.274133 & -0.321746 \\ 0.211205 & -0.523083 & 0.311878 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \quad (4.32)$$

$I$  大约在橙-蓝方向,  $Q$  大致与紫-绿方向对应。

图 4-19 显示了上面提到的彩色图像分解为  $YIQ$  分量的效果。这里只显示  $I$  和  $Q$  分量, 因为  $Y'$  分量与图 4-19 中的相同。

对于这幅图像, 大多数能量都在  $Y'$  分量中被捕获, 它是很典型的。然而, 在这种情况下,  $YIQ$  分解对于形成图像的层次序列更有帮助: 对于 8 位  $Y'$  分量, 均方根 (Root-Mean-Square, RMS) 值为 137 (255 为最大可能值)。  $U$ 、 $V$  分量具有均方根值 43 和 44。对于  $YIQ$  分解,  $I$  和  $Q$  分量具有均方根值 42 和 14, 所以它们更好地区分了颜色的优先次序。最初, NTSC 分配 4.2MHz 给  $Y$ , 1.5MHz 给  $I$ , 0.6MHz 给  $Q$ 。现在,  $I$  和  $Q$  都得到了 1.0MHz。

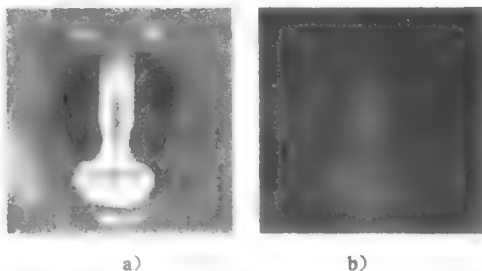


图 4-19 彩色图像的  $I$  和  $Q$  分量

105  
106

#### 4.3.4 YCbCr 颜色模型

分量 (三路信号, 量化) 数字视频的国际标准是官方的推荐 ITU-R BT.601-4 (也称为 “Rec.601”)。这个标准使用另一个颜色空间 YCbCr。YCbCr 变换用于 JPEG 图像压缩和 MPEG 视频压缩, 与 YUV 变换紧密相关。YUV 通过调整,  $C_b$  成为  $U$ , 但是系数为 0.5 乘以  $B'$ 。在一些软件系统中,  $C_b$  和  $C_r$  也被移动, 这样其值在 0~1 之间。这就产生出下面的等式:

$$\begin{aligned} C_b &= ((B' - Y')/1.772) + 0.5 \\ C_r &= ((R' - Y')/1.402) + 0.5 \end{aligned} \quad (4.33)$$

将其展开, 我们有

$$\begin{bmatrix} Y' \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \\ 0.5 \end{bmatrix} \quad (4.34)$$

然而, 在实践中, Rec.601 指定 8 位编码,  $Y'$  的最大值仅为 219, 最小值为 +16。小于 16 和大于 235 的值, 表示为头上空间 (headroom) 与脚下空间 (footroom), 保留起来用于其他用途。 $C_b$  和  $C_r$  的变化范围为  $\pm 112$ , 偏移值为 +128 (换句话说, 最大值为 240, 最小值为 16)。如果  $R'$ 、 $G'$ 、 $B'$  是  $[0, +1]$  间的浮点数, 我们通过变换 [9] 得到  $[0..255]$  的  $Y'$ 、 $C_b$  和  $C_r$ :

$$\begin{bmatrix} Y' \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \quad (4.35)$$

实际上, 输出范围也被限制在  $[1, 254]$ , 因为 Rec.601 同步信号通过编码 0~255 给出。

#### 4.4 进一步探索

更进一步, 颜色是人类的喜好之一, 并且它是使多媒体如此引人注目的重要属性之一。一般说来, 颜色领域最常用的参考是经典手册 [2]。今天使用的重要技术纲要文集 [10]。

本网站上第 4 章的 Further Exploration 部分的链接包括:

107

- WWW 上介绍伽马校正详情的出版物。
- 用于 WWW 应用的新的 sRGB 标准颜色空间的完整规范。
- 颜色变换的精彩回顾和标准颜色 FAQ。
- 练习颜色变换函数的 MATLAB 脚本(它是 MATLAB Image Toolbox 的一部分): 将标准 Lena 图像转换为 YIQ 和 YCbCr。
- 新的颜色空间。新的 MPEG 标准 MPEG-7 (第 12 章讨论) 在一定程度上回避了棘手的问题, 即通过包含六个颜色空间在一个标准定义中使用哪个颜色空间。它们中的一个 HSV 空间的变体——HMMD 颜色空间, 它支持一个简单的颜色量化(从 24 位下降到 8 位颜色), 即它相当于复杂的向量颜色量化(比如, 考虑把一幅图像中的颜色更仔细但是代价更高地映射到颜色查找表中)。这个新的颜色空间也许会变得重要。

## 4.5 练习

### 1. 解释如下与颜色相关的术语集合:

- 波长
- 颜色等级
- 亮度
- 白度

把下面的(更加模糊地陈述的)特性与上面的术语对应起来。

- 亮度
- 色调
- 饱和度
- 色度

### 2. 什么颜色是户外颜色? 举个例子, 你认为哪个波长的峰值能量对应红色的日落? 对于蓝色天空的光线呢?

### 3. “LAB 色域涵盖了可见光谱中所有的颜色。”

- 这句话是什么意思? LAB 与颜色如何联系? 简要描述即可。
- LAB 色域、CMYK 色域和显示器色域的大致相关尺寸是多少?

### 4. 图 4-11 中色度的马脚曲线来源于哪里? 我们能够计算它吗? 对找到“光谱轨迹”的问题写一段简短的伪代码来解决。提示: 图 4-20a 显示了图 4-10 的颜色匹配函数, 它是用三维空间中的点集来画的。图 4-20b 显示了这些点映射到另一个三维点集。另一个提示: 试着编程解决这个问题, 这样有助于你更明白地解答。

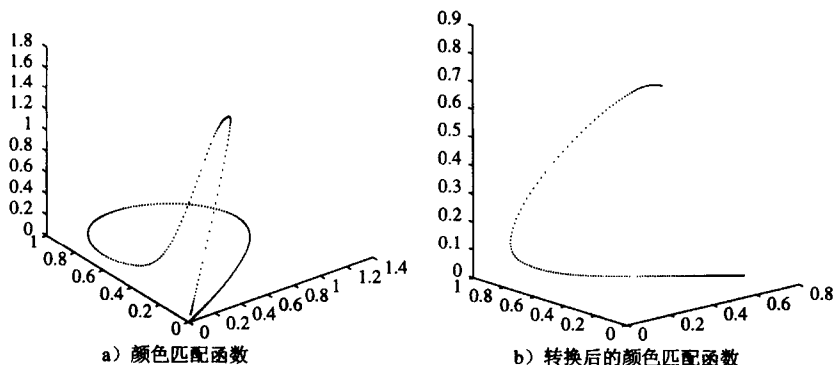


图 4-20

5. 假设我们使用一个新的颜色匹配函数集合  $\bar{x}^{new}(\lambda)$ 、 $\bar{y}^{new}(\lambda)$ 、 $\bar{z}^{new}(\lambda)$ ，其值为：

$\lambda$ (nm)	$\bar{x}^{new}(\lambda)$	$\bar{y}^{new}(\lambda)$	$\bar{z}^{new}(\lambda)$
450	0.2	0.1	0.5
500	0.1	0.4	0.3
600	0.1	0.4	0.2
700	0.6	0.1	0.0

在这个系统中，等能量白光  $E(\lambda)$  的色度值  $(x, y)$  是什么？这里，对所有波长  $\lambda$ ， $E(\lambda) \equiv 1$ 。请解释。

6. (a) 假设图像没有用便携式摄像机进行伽马校正。一般来说，它在屏幕上显示的效果如何？  
 (b) 如果对于存储的图像像素，我们人为增加输出伽马值（我们可以借助 Photoshop 做到这一点），将会发生什么？对图像会产生什么影响？
7. 假设图像文件的值在每个颜色通道都是 0~255。如果对于红色通道，我们定义  $\bar{R} = R/255$ ，希望通过传入一个新的值  $\bar{R}' \approx \bar{R}^{1/2.0}$  到显示设备来执行伽马校正。通常用整数来操作。假设我们通过创造 0~255 间的新的整数值来近似计算：

$$(int)(255 \cdot (\bar{R}^{1/2.0}))$$

- (a) 评论（粗略地）一下这种操作对实际上可使用的显示级别有何影响？提示：用某种语言编写代码将有助于你更好地理解这种机制，并且能让你容易地数出输出级别。  
 (b) 0~255 级别的哪一端更多地受到伽马校正的影响，是低端（靠近 0）还是高端（靠近 255）？为什么？每一端受到多大影响？

8. 在许多计算机图形学应用中，伽马校正仅在颜色查找表中使用。给出使用伽马校正的颜色查找表的头五项内容。  
 提示：编写代码将省去你使用计算器的麻烦。

9. 设计一个程序来产生图 4-21，显示符合 SMPTE 规范的显示器的色域。

10. 色调是与亮度和多少纯白被加入无关的颜色。我们能够简单定义色调为  $R:G:B$  的比例集合。假设一种颜色（比如，一种 RGB）被 2.0 除，这样，RGB 三元组是它以前值的 0.5 倍。使用数字值解释：

- (a) 如果伽马校正是在除以 2.0 之后颜色被存储之前使用，如 CRT 显示设备发出的光线的  $R:G:B$  比例相同，那么更深的 RGB 是否具有与原来相同的色调？

（我们不讨论改变我们感知的任何心理学影响，这里仅关心机器本身。）

- (b) 如果不采用伽马校正，当显示时，第二个 RGB 是否与第一个 RGB 有相同的色调？  
 (c) 对于什么颜色三元组，色调总是不改变？

11. 我们希望制作一个令人满意的易读图形。假设我们把背景颜色设为粉红，那么前端文字应当使用什么颜色才能够最易读？证明你的答案。

12. 为了使最后的打印变得简单，我们买了一个装有 CMY 传感器（与 RGB 传感器相反）的照相机（CMY 照相机是实际可用的）。

- (a) 大致画出反映这种照相机的灵敏度与频率的光谱曲线。  
 (b) CMY 照相机的输出能够用来生成普通的 RGB 图片吗？为什么？

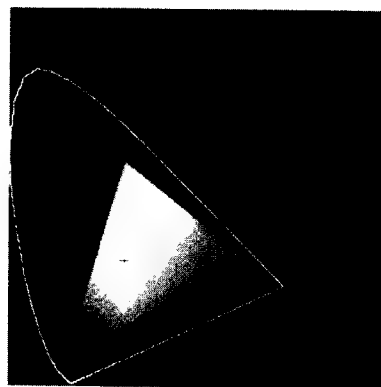


图 4-21 SMPTE 监视器色域  
 （彩色插页中也有此图）

13. 彩色喷墨打印机使用 CMY 模型。当青色墨水喷洒在一片白纸上时:

- (a) 为什么青色看起来像是在日光下呢?
- (b) 在蓝色光线下它看起来像什么颜色? 说明原因。

#### 4.6 参考文献

- [1] D.H. Pritchard, "U.S. Color Television Fundamentals — A Review," *IEEE Trans. Consumer Electronics* 23(4): p. 467–478, 1977.
- [2] G. Wyszecki and W.S. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulas*, 2nd ed., New York: Wiley, 1982.
- [3] R.W.G. Hunt, "Color Reproduction and Color Vision Modeling," in *1st Color Imaging Conference: Transforms & Transportability of Color*, Society for Imaging Science & Technology (IS&T)/Society for Information Display (SID) joint conference, 1993, 1–5.
- [4] M.J. Vrhel, R. Gershon, and L.S. Iwan, "Measurement and Analysis of Object Reflectance Spectra," *Color Research and Application*, 19: 4–9, 1994.
- [5] R.W.G. Hunt. *The Reproduction of Color*, 5th ed., Tolworth, Surry, U.K.: Fountain Press, 1995.
- [6] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice in C*, 2nd ed., Reading MA: Addison-Wesley, 1996.
- [7] Mark D. Fairchild, *Color Appearance Models*, Reading MA: Addison-Wesley, 1998.
- [8] D. Travis, *Effective Color Displays*, San Diego: Academic Press, 1991.
- [9] C.A. Poynton, *A Technical Introduction to Digital Video*, New York: Wiley, 1996.
- [10] P. Green and L. MacDonald, eds., *Colour Engineering: Achieving Device Independent Colour*, New York: Wiley, 2002.

## 第5章 视频中的基本概念

在本章中，我们将介绍和视频相关的基本概念。数字视频压缩技术将在第10章~第12章单独讨论。

我们首先介绍视频的如下几个方面的内容，以及它们是如何对多媒体应用产生影响的：

- 视频信号的类型
- 模拟视频
- 数字视频

由于视频信号是由多种信号源所产生的信号组成，因此我们首先从信号本身开始介绍。模拟视频是用连续的（时变的）信号表示的，本章的第一部分将就模拟视频如何量化的问题展开讨论。数字视频是用一系列数字图像表示的，本章的第二部分将就相关标准和规范（例如 HDTV）展开讨论。

### 5.1 视频信号的类型

视频信号可以分为三类：分量视频(Component video)、复合视频(Composite video)和 S-video。

#### 5.1.1 分量视频

高端视频系统（例如视频工作室）分别使用三路视频信号来表示红、绿、蓝三种图像平面。这类视频称为分量视频。这种系统有三根线和连接器，将照相机或其他设备同电视或显示器连接起来。

颜色信号并不局限在 RGB 分色上。正如在第4章中所讲到的，我们可以通过对 RGB 信号的亮度-色度变换来生成三种信号——例如采用 YIQ 或者 YUV 颜色空间变换。相比之下，大多数计算机系统采用分量视频，该视频包含 RGB 信号的分离信号。

对于分色系统来说，由于在这三种不同的信道之间没有任何色度、亮度干扰（不同于复合视频和 S-video 视频），所以分量视频的颜色再现能力最好。但是，分量视频需要更多的带宽和三种分量间良好的同步机制。

112

#### 5.1.2 复合视频

在复合视频中，颜色（色度）信号和强度（亮度）信号混合成一个的载波。色度是由两种颜色分量（ $I$  和  $Q$ ，或者  $U$  和  $V$ ）构成的。复合视频用于彩色电视广播，并且它是兼容黑白电视广播的。

在 NTSC 电视中（例如在[1]中）， $I$  和  $Q$  合并为色度信号，颜色副载波频率将色度信号移与亮度信号共享的信道的高频段。色度和亮度分量可以在接收端进行分离，并且这两种颜色分量可以进一步地恢复。

当连接到电视或者 VCR 上时，复合视频仅用一根线（和一个连接器，例如共轴电缆两端安装的 BNC 连接器或者普通电线两端的 RCA 插槽），而且视频颜色信号是混合的，而不是分开发送的。音频信号可以附加到这样的信号上传送。由于颜色信息已经混合起来，颜色和强度都封装到同一个信号中，那么亮度和色度之间的干扰就不可避免。

### 5.1.3 S-Video

作为折中方案，S-video（分离视频或超视频，例如在 S-VHS 中）使用两条电线，一条用于亮度信号，另一条用于混合的色度信号。这样，颜色信息与关键的灰度信息之间的色度亮度干扰会少一些。

将亮度作为信号，一部分原因是对于视觉感知来说黑白信息是至关重要的。正如前面几章所讲的，人类对灰度图中的空间分辨率更为敏感，彩色图的彩色部分（相对于黑白部分）则稍差。因此，在发送过程中，颜色信息就没有强度信息那样精确。由于我们只看到较大的颜色块，因而就可以发送较少的颜色细节信息。

## 5.2 模拟视频

大多数电视是通过模拟信号收发电视节目的。当电子信号被接收时，我们可以假设由于伽马校正（详见 4.1.6 节）的作用，如果亮度不是电压的线性函数的话，至少也是电压的单调函数。

模拟信号  $f(t)$  对时变的图像进行采样。所谓的渐进扫描按照时间间隔逐行跟踪完整的图像（帧）。高分辨率的计算机显示器的时间间隔一般为  $1/72$  秒。

在电视、显示器或多媒体标准中，采用隔行扫描（interlaced scanning）。隔行扫描先扫描奇数行，然后扫描偶数行。这样就产生“奇数域”和“偶数域”，两个域组成一帧图像。

事实上，奇数行（从第一行开始）扫描到奇数域中最后一行的中间部位，偶数行扫描就从第一行的中线位置开始。图 5-1 显示了这种方案。

首先扫描奇数行（用实线表示），从 P 到 Q，然后从 R 到 S 等，最后到 T 终止；接下来偶数行扫描从 U 开始，到 V 停止。扫描线在一个微小电压的作用下不是水平的，随着时间的推移向下缓慢移动电子束。

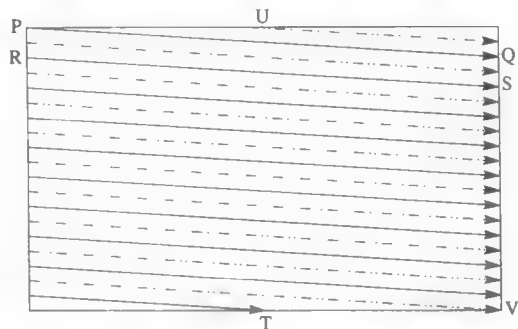


图 5-1 隔行光栅扫描

发明隔行扫描技术是为了解决定义标准之后，很难以足够快的速度（避免图像闪烁）传送一帧图像信息的问题。当我们眼睛看到双倍扫描图像时可以减少感觉到的闪烁。

由于隔行扫描的作用，奇数行和偶数行交替显示。一般情况下，我们感觉不到这种交替过程，除非在屏幕上有快速动作发生时，可能产生模糊的图像。例如，在图 5-2 中显示的视频，移动的直升机就比静止的背景要模糊一些。

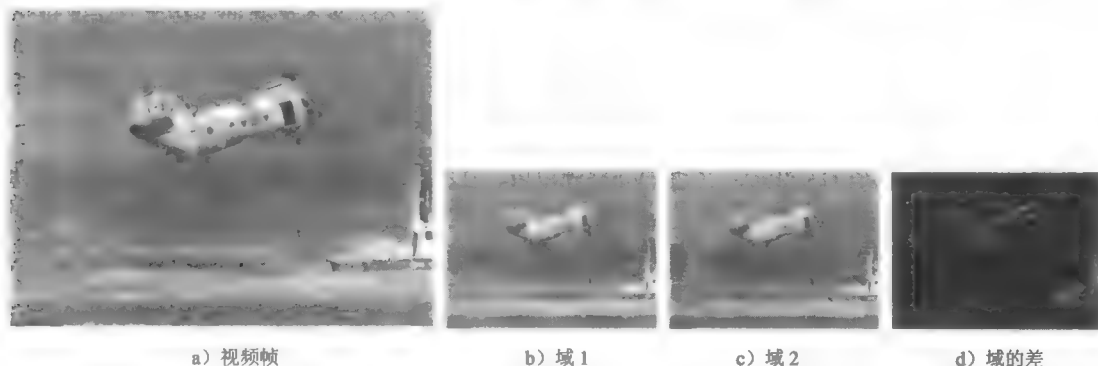


图 5-2 隔行扫描为每帧产生 2 个域

有时候我们需要改变帧速率、缩放视频甚至从应用隔行扫描的视频源中产生停止图像，因此还要采用各种方案去除交错模式。去除交错模式最简单的一种方法就是摒弃其中一个扫描域，复制另一个域的扫描线，这样导致其中一个域的信息完全丢失。其他更复杂的方法都是保留两个域的信息。

CRT 显示由类似荧光灯的东西构成，为保证图像平滑必须每秒闪动 50~70 次。在欧洲，这种设计是和电力系统 50Hz 的频率紧密相关的，他们以每秒 25 帧（即 25fps）的频率进行数字化的视频。在北美，电力系统采用 60Hz 的频率，则以 30fps 的帧速率进行数字化。

图 5-1 中从 Q 到 R 的跳变称为水平回扫（horizontal retrace），在此过程中 CRT 的电子束是空白的。从 T 到 U 或者从 V 到 P 的跳变被称为垂直回扫（vertical retrace）。

由于电压是一维的（它仅随时间变化而改变），那么我们怎么知道什么时候新的视频帧开始呢？也就是说，电子信号中哪部分能够告诉我们必须从屏幕左侧开始重新扫描？

模拟视频中采用的一种解决方案是用零起点的微小偏移电压表示“黑”，另一个电压（例如零电压）表示一行扫描的起始。也就是说，我们可以采用“比黑色更黑”（blacker-than-black）零信号表示一行扫描的起始。

图 5-3 显示一个典型的 NTSC 中复合视频扫描线的电子信号。“白”线处电压峰值为 0.714V；“黑”线为稍高于零电压的 0.055V；而“消隐”线才用零电压表示。如图所示，信号中消隐脉冲时间段主要用于同步，同步信号大致为 -0.286V 左右。事实上，可靠同步是非常重要的，以至于一些特殊的信号中使用信号的 30% 来处理同步。

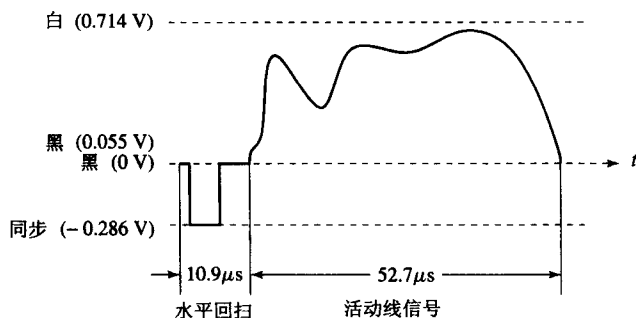


图 5-3 NTSC 中扫描线的电子信号

垂直回扫和同步的思想与水平回扫非常类似，只是它仅仅在每个扫描域中出现一次。Tekalp[2] 中详细讨论模拟视频和数字视频的内容。手册[3]中深入探讨了视频处理中的一些基本问题。

### 5.2.1 NTSC 视频

NTSC TV 的标准主要在北美和日本使用。它采用 4:3 的画面比例（也就是图像的长宽比），每秒 30 帧，每帧 525 条扫描线。

更精确地说，由于历史原因，NTSC 采用每秒 29.97 帧，换句话说，每帧 33.37ms。NTSC 遵循隔行扫描系统的设计，每一帧分为两个域，每个域有 262.5 行。因此，水平扫描率是  $525 \times 29.97 \approx 15734$  行/秒，每行扫描时间是  $1/15734 \approx 63.6 \mu\text{s}$ 。其中水平回扫占用  $10.9 \mu\text{s}$ ，这就剩下  $52.7 \mu\text{s}$  用于活动视频线信号，这段时间用于显示图像数据（如图 5-3 所示）。

图 5-4 显示 NTSC 视频光栅中“垂直回扫与同步”和“水平回扫与同步”的效果。消隐信息放置于每个域起始位置为控制信息预留的 20 行中。这样，每帧中活动视频线（active video line）的数量只有 485 条。类似地，光栅左侧差不多 1/6 的部分是空白的，用于水平回扫和同步。非空

白的像素称为活动像素 (active pixel)。

像素通常落于扫描线之间。因此, NTSC TV 即使采用非隔行扫描也仅能显示 340 (可视) 行, 大致是 485 条特定的活动视频线的 70% 左右。采用隔行扫描, 该数值只能达到 50% 左右。

图像数据在消隐区域时不能编码, 但是其他的信息可以放置到该区域中, 例如 V-chip 信息、立体声信道数据和用不同语言显示的字幕等。

NTSC 视频是一种没有固定水平分辨率的模拟信号。因此, 我们必须预先确定对显示信号采样的次数。每次采样对应于一个像素输出。点时钟 (pixel clock) 将视频的每个水平行划分为采样。点时钟的频率越高, 每行的采样点就越多。

不同的视频格式有不同的行采样点个数, 如表 5-1 所示。激光盘大致和 Hi-8 相同。(相比之下, 用于数字视频的 miniDV 的 1/4 英寸磁带带有 480 行, 每行 720 个采样点。)

NTSC 采用 YIQ 颜色模型。我们应用正交调制的技术来整合  $I$  (同相) 信号和  $Q$  (正交) 信号为一个单独的色度信号  $C[1, 2]$ :

$$C = I \cos(F_{sc}t) + Q \sin(F_{sc}t) \quad (5.1)$$

调制的色度信号也就是我们所知的颜色副载波 (color subcarrier), 其幅值为  $\sqrt{I^2 + Q^2}$ , 其相位为  $\tan^{-1}(Q/I)$ 。 $C$  的频率为  $F_{sc} \approx 3.58\text{MHz}$ 。

如式 (5.1) 所示, 时域上  $I$  和  $Q$  信号要与和频率  $F_{sc}$  有关的余弦函数和正弦函数相乘。该式相当于在频域上对  $F_{sc}$  和  $-F_{sc}$  两处脉冲函数作用下的傅里叶变换进行卷积运算。结果, 一份  $I$  和  $Q$  的频率谱分别集中在  $F_{sc}$  和  $-F_{sc}$  处。

NTSC 的复合信号是亮度信号  $Y$  和色度信号的复合信号, 定义如下:

$$\text{复合信号} = Y + C = Y + I \cos(F_{sc}t) + Q \sin(F_{sc}t) \quad (5.2)$$

NTSC 根据人类对于颜色的细节内容 (颜色变化的高频部分) 不敏感的特性, 给予  $Y$  的带宽为 4.2MHz, 而  $I$  只有 1.6MHz,  $Q$  只有 0.6MHz。如图 5-5 所示, 在 NTSC 视频信道中, 图像载波信号在 1.25MHz 处, 该信道的总带宽有 6MHz。色度信号通过  $F_{sc} \approx 3.58\text{MHz}$  的频率携带到信道的高频段, 因此集中在  $1.25+3.58=4.83\text{MHz}$  处。由于  $Y$  的高频部分的幅值比其低频部分的幅值小得多, 这样就在很大程度上减小了  $Y$  (亮度) 和  $C$  (色度) 之间的干扰。

此外, 正如 Blinn[1]所解释的, 应该格外注意将离散的  $Y$  和  $C$  频谱交错起来, 这样能够进一步减少它们之间的干扰。图 5-5 说明了交错过程, 这里  $Y$  的频率分量 (从傅里叶变换得到) 用实线表示,  $I$  和  $Q$  分量用虚线表示。结果  $Y$  的 4.2MHz 频率带与  $I$  的 1.6MHz 带和  $Q$  的 0.6MHz 带交叠在一起。

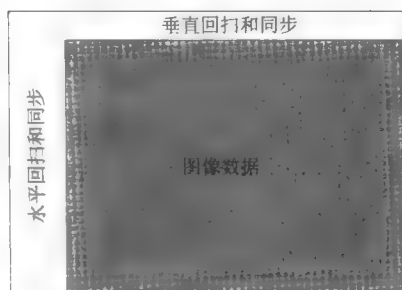


图 5-4 包括回扫和同步数据的视频光栅

表 5-1 各种模拟视频格式中行采样点

格 式	每行采样数
VHS	240
S-VHS	400~425
Beta-SP	500
Standard 8 mm	300
Hi-8 mm	425

⊖  $-F_{sc}$  是傅里叶变换中的一个数学概念。在物理频谱中, 只使用正频率。



在接收端对复合信号解码的第一步是分离  $Y$  和  $C$ 。一般而言,低通滤波器可以用于提取  $Y$  分量,该滤波器置于信道的低频端。高质量画面的电视机也采用根据  $Y$  和  $C$  交错的原理制成的梳状滤波器[1]。

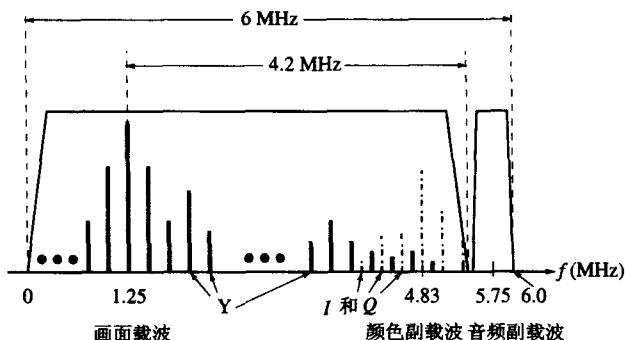


图 5-5 NTSC 频谱中交错的  $Y$  和  $C$  信号

分离  $Y$  分量之后,通过解调色度信号  $C$  分别提取出  $I$  和  $Q$  分量。

为了提取  $I$  分量,要进行以下工作:

1) 信号  $C$  乘以  $2\cos(F_{sc}t)$ 。

$$\begin{aligned} C \cdot 2\cos(F_{sc}t) &= I \cdot 2\cos^2(F_{sc}t) + Q \cdot 2\sin(F_{sc}t)\cos(F_{sc}t) \\ &= I \cdot (1 + \cos(2F_{sc}t)) + Q \cdot 2\sin(F_{sc}t)\cos(F_{sc}t) \\ &= I + I \cdot \cos(2F_{sc}t) + Q \cdot \sin(2F_{sc}t) \end{aligned}$$

2) 应用低通滤波器得到  $I$  分量,并摒弃两个高频项 ( $2F_{sc}$ )。

类似地,首先将  $C$  乘以  $2\sin(F_{sc}t)$ , 然后进行低通滤波就可以得到  $Q$  分量。

NTSC 中的 6MHz 带宽是很紧张的。音频副载波频率是 4.5MHz, 它将音频带宽中心移至  $1.25+4.5=5.75\text{MHz}$  处(如图 5-5 所示)。实际上,这与颜色副载波的距离太近了,这就是音频信号和颜色信号之间产生干扰的一个潜在原因,而这是由于 NTSC 彩色电视将帧速率降低为  $30 \times 1000/1001 \approx 29.97\text{fps}$  造成的[4]。结果,目前采用的 NTSC 颜色副载波频率稍微降低了一点:

$$f_{sc} = 30 \times 1000 / 1001 \times 525 \times 227.5 \approx 3.579545 \text{ MHz}$$

这里的 227.5 是在 NTSC 广播电视中采用的每条扫描线的颜色采样点的个数。

## 5.2.2 PAL 视频

PAL (Phase Alternating Line, 逐行倒相) 标准是由德国科学家设计的 TV 标准。它采用每帧 625 条扫描线,每秒 25 帧(即 40 毫秒/帧)的帧率,4:3 的画面比例和隔行扫描域。它的广播 TV 信号也在复合视频中采用。PAL 标准在西欧、中国、印度和许多其他国家广泛使用。

PAL 采用 YUV 颜色模型,其信道宽度为 8MHz,其中 5.5MHz 分配给  $Y$  分量,  $U$  和  $V$  分量各分配 1.8MHz。颜色副载波频率为  $f_{sc} \approx 4.43 \text{ MHz}$ 。为了提高画面质量,色度信号在连续的扫描线中使用交叠符号(例如  $+U$  和  $-U$ ),也就是逐行倒相这一名字的由来<sup>①</sup>。这样便于在接收端使用梳状滤波器,作用是平均化连续扫描行中的信号,这样就不用色度信号(总有负号)来分离  $Y$  和  $C$  分量并能获得高质量的  $Y$  信号。

① 参见 Blinn[1], NTSC 为每个扫描行选择一个半整数(227.5)的颜色采样,因而具色度信号也在相连的扫描行变换符号。

### 5.2.3 SECAM 视频

SECAM 是由法国科学家设计的第三大广播电视标准。SECAM (System Electronique Couleur Avec Memoire) 表示顺序传送与存储彩色电视系统。SECAM 也是采用每帧 625 条扫描线, 即每秒 25 帧的帧率, 4:3 的尺寸比和隔行扫描域。最初的设计要求有更多数目的扫描线 (800 条以上), 但是最终版本只有 625 条。

SECAM 和 PAL 十分类似的, 只是在颜色编码方案上有所区别。在 SECAM 中,  $U$  和  $V$  信号通过分别在 4.25MHz 和 4.41MHz 处使用单独的颜色副载波来进行调制。它们是用交替的线发送的, 也就是说, 在每条扫描线中,  $U$  和  $V$  信号只有一个被发送。

表 5-2 给出三种主要的模拟广播电视系统的对比。

表 5-2 模拟广播电视系统的对比

电视制式	帧率 (fps)	扫描线数	信道总宽度 (MHz)	分配宽度 (MHz)		
				$Y$	$I$ 或 $U$	$Q$ 或 $V$
NTSC	29.97	525	6.0	4.2	1.6	0.6
PAL	25	625	8.0	5.5	1.8	1.8
SECAM	25	625	8.0	6.0	2.0	2.0

## 5.3 数字视频

视频的数字显示技术的优点有很多。它允许

- 在内存或者数字设备上存储视频以便于进一步的处理 (去噪、剪切和粘贴等操作) 以及集成到各种各样的多媒体应用程序中。
- 直接访问, 这样使得非线性视频编辑更加简单。
- 重复记录而不降低图像的质量。
- 便于加密, 对信道噪声的容忍度更高。

119

在 Sony 或 Panasonic 的早期录像机中, 数字视频是由复合视频构成的。一般来说, 现代的数字视频都采用分量视频, 尽管 RGB 信号首先被转换成某些颜色分量空间, 例如 YUV 空间。通常的颜色空间是 YCbCr[5]。

### 5.3.1 色度的二次采样

由于人类对彩色没有黑白那样敏感, 因此消减色度信号是很有意义的。我们使用一些有趣但未必有教益的名称来标示不同的方案。首先, 用数字表示每 4 个原始像素中多少像素值会实际发送。因此, 色度二次采样方案 “4:4:4” 表示没有使用色度二次采样。每个像素的  $Y$ 、 $Cb$  和  $Cr$  值被传送, 每个  $Y$ 、 $Cb$  和  $Cr$  都是 4。

方案 “4:2:2” 表示  $Cb$  和  $Cr$  信号是因子为 2 的水平二次采样。也就是说,  $Y$  中水平方向上 4 个像素标为 0~3 的  $Y$  发送出去, 同时每隔一个的  $Cb$  和每隔一个的  $Cr$  也一起发送出去。例如 ( $Cb_0$ ,  $Y_0$ ) ( $Cr_0$ ,  $Y_1$ ) ( $Cb_2$ ,  $Y_2$ ) ( $Cr_2$ ,  $Y_3$ ) ( $Cb_4$ ,  $Y_4$ ) 等等。

方案 “4:1:1” 水平二次采样的因子是 4。方案 “4:2:0” 在水平方向和垂直方向都进行二次采样, 其因子是 2。从理论上说, 平均的色度像素置于行和列之间, 如图 5-6 所示。我们可以看到方案 4:2:0 实际上是另一种 4:1:1 的模式, 其中每 4 个像素我们发送 4, 1, 1 个值。因此, 这种标示的方法不是很可信的助记方法。

其中方案 4:2:0 与其他配合被广泛地应用到 JPEG 和 MPEG (在第二部分中详细讲述) 中。

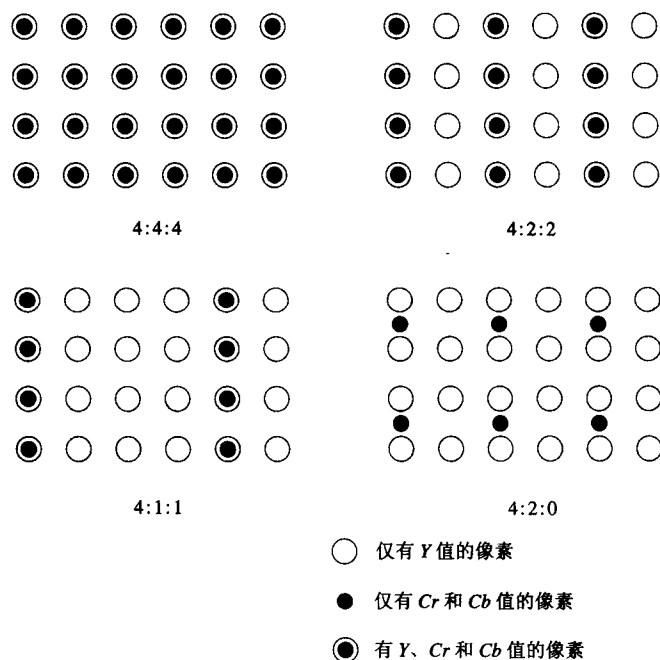


图 5-6 色度的二次采样

### 5.3.2 数字视频的 CCIR 标准

CCIR 是国际无线电咨询委员会（Consultative Committee for International Radio）的简称。它制定的重要标准之一就是针对于分量数字视频（4.3.4 节介绍过）的 CCIR-601 标准。该标准目前已经成为 ITU-R-601 标准——一个针对专业视频应用的国际标准。它已经被某些数字视频格式（包括十分流行的 DV 视频）所采用。

NTSC 版本中包含有 525 条扫描线，每条扫描线有 858 个像素（其中 720 个像素可见，不在消隐周期中）。因为 NTSC 版本中使用 4:2:2 方案，每个像素能够用两个字节表示（8 位用于表示 Y，8 位交替表示 Cb 和 Cr）。这样，CCIR 601（NTSC）的数据率（包括消隐和同步数据，不包括音频数据）大约是 216Mbps（每秒兆位）：

$$525 \times 858 \times 30 \times 2 \text{ 字节} \times 8 \frac{\text{位}}{\text{字节}} \approx 216 \text{ Mbps}$$

在消隐过程中，数字视频系统可使用额外的数据容量来传送音频信号，外文译文或者纠错信息。

表 5-3 显示一些数字视频规范，其中所有规范的画面比例是 4:3。CCIR 601 标准使用的是隔行扫描，这样每个域中只有垂直分辨率的一半（例如在 NTSC 中是 240 行）。

表 5-3 数字视频规范

	CCIR 601 525/60 NTSC	CCIR 601 625/50 PAL/SECAM	CIF	QCIF
亮度分辨率	720×480	720×576	352×288	176×144
色度分辨率	360×480	360×576	176×144	88×72
颜色二次采样	4:2:2	4:2:2	4:2:0	4:2:0

(续)

	CCIR 601 525/60 NTSC	CCIR 601 625/50 PAL/SECAM	CIF	QCIF
画面比例	4:3	4:3	4:3	4:3
场/秒	60	50	30	30
是否隔行扫描	是	是	否	否

CIF 是公共中间格式 (Common Intermediate Format) 的简称, 该规范是由国际电报电话咨询委员会 (International Telegraph and Telephone Consultative Committee, CCITT) 制定的, 现在 CCITT 由国际电信同盟 (International Telecommunication Union) 代替, 该组织同时监管在同一个国际团体中的电信标准化部门 (ITU-T) 和无线电通信部门 (ITU-R)。CIF 的思想与 VHS 质量的思想大致一样的, 用于为低码率的情况制定格式规范。CIF 使用顺序扫描 (非隔行扫描) 的方式。QCIF 表示 Quarter-CIF, 主要应用于均匀的低码率的情况。所有的 CIF 或 QCIF 分辨率都被 8 整除, 除 88 外所有的情况均能被 16 整除。这样便于 H.261 和 H.263 协议中基于块的视频编码, 这部分内容将在第 10 章中讨论。

CIF 是 NTSC 和 PAL 的折中方案, 因为它既采用 NTSC 的帧速率, 又采用 PAL 中活动扫描线个数的一半。当在电视机中播放节目时, 基于 NTSC 的电视节目首先要转换扫描线的个数, 而基于 PAL 的电视节目就需要改变帧率的大小。

### 5.3.3 HDTV

宽屏幕电影不同于常规的电影, 它允许观众靠近屏幕并有一定程度上的互动参与。宽屏幕电影可以看到较宽的区域, 特别是引入视觉边限的技术, 让人身临其境。高清电视 (HDTV) 不是增加每个单元区域的清晰度, 而是增加可视域, 特别是宽度。

第一代 HDTV 是基于 20 世纪 70 年代后期日本的 Sony 和 NHK 开发的模拟技术。在日本, HDTV 成功地播放了 1984 年洛杉矶奥运会。MUSE (Multiple sub-Nyquist Sampling Encoding) 是改进的 NHK 的 HDTV 系统, 该系统采用混合的模拟/数字技术, 并在 20 世纪 90 年代得到应用。它采用 1125 条扫描线隔行扫描, (每秒 60 个域), 画面比例为 16:9。它使用卫星广播电视节目, 该方式很适合日本国情, 仅仅使用一个或者两个卫星即可覆盖日本全国。使用的直播卫星 (Direct Broadcast Satellite, DBS) 信道有 24MHz 的带宽。

122

总之, 陆地广播、卫星广播、光缆和宽带网络是传送 HDTV 和传统的 TV 节目的一些可行的方法。因为非压缩的 HDTV 要求 20MHz 的带宽, 这并不适合当前 6MHz 或者 8MHz 的信道带宽, 所以各种压缩技术正在研究中。我们可以预见, 高质量的 HDTV 信号即使经过压缩, 也要通过多个信道进行传送。

1987 年, FCC 决定 HDTV 标准必须和现有的 NTSC 标准兼容, 必须限制在现有的 VHF 和 UHF 波段内。到 1988 年底, 北美提出一系列相关方案, 全部都是基于模拟的或者模拟/数字混合的。

1990 年, FCC 提出另一种提案, 它更适合于全分辨率的 HDTV。它们决定 HDTV 可以同时用现有的 NTSC 电视广播, 最终再代替它。数字 HDTV 的开发立刻在北美飞速发展。

借鉴数字 HDTV 的各个提案, 1993 年, FCC 作出了一个重要决定。由 General Instruments、MIT、Zenith 和 AT&T 等机构, Thomson、Philips、Sarnoff 等人联合组建 Grand Alliance 组, 并提出四个主要提案。它后来发展成为 ATSC, ATSC 主要负责 HDTV 的电视广播的相关标准的制定。1995 年, 美国的 FCC 咨询委员会关于 Advanced Television Service 建议应该采用 ATSC 的数字电视标准。

该标准支持表 5-4 中所示的视频扫描格式。在表中, I 表示隔行扫描, P 表示渐进 (非隔行) 扫描。支持整数帧速率和 NTSC 帧速率, 也就是 60.00 fps 或 59.94 fps, 30.00 fps 或 29.97 fps, 24.00 fps 或 23.98fps。

表 5-4 ATSC 支持的高级数字视频格式

每线的活动像素数	活动线数	画面比例	画面率
1920	1080	16:9	60I 30P 24P
1280	720	16:9	60P 30P 24P
704	480	16:9 和 4:3	60I 60P 30P 24P
640	480	4:3	60I 60P 30P 24P

对于视频而言, MPEG-2 被选为压缩标准。正如第 11 章将要讲到的, 它采用 MPEG-2 中的 Main Profile 的 Main Level 到 High Level。对于音频而言, AC-3 是相关标准。它支持 5.1 信道杜比环绕立体声, 即五个环绕立体声信道加上低音炮信道。

一般的 TV 和 HDTV 的显著区别[4, 6]就在于后者有 16:9 的画面比例而前者只有 4:3 (事实上, HDTV 比一般的 TV 宽 1/3)。HDTV 的另一个特点是它朝着渐进扫描 (非隔行扫描) 的方向发展。原因是隔行扫描会给运动物体引入锯齿边, 给水平边带来摇摆。

FCC 计划在 2006 年以前将所有模拟信号的广播服务替换成数字电视广播。模拟电视机用户将可以通过一个 8-VSB (8-level vestigial sideband) 解调器来接收该信号。这项服务将包括:

- 标清数字电视——当前的 NTSC 制式电视或更高。
- 增强清晰度电视——480 根活动线或者更多, 即表 5-4 中的第三行和第四行。
- 高清电视——720 根活动线或者更多。到现在为止, 最常用的选择是 720P (720 线, 渐进扫描, 30fps) 和 1080I (1080 线, 隔行扫描, 30fps 或 60fps)。后者比前者能提供稍微好一些的图像质量, 但是对带宽要求更高。

## 5.4 进一步探索

Tekalp[5]涵盖数字视频技术中的一些重要问题。Steinmetz 和 Nahrstedt[7]在他们的著作的第 5 章中对视频和电视系统做了详细的讨论。Poynton[6]提供了数字视频和 HDTV 的更新的介绍。

本书网站上与本章中有关的链接包括:

- NTSC 电视的教程。
- ATSC 官方网站。
- 数字电视的前沿消息。
- HDTV 的介绍。
- FCC 官方网站。

## 5.5 练习

1. NTSC 视频每帧 525 线, 每线  $63.6 \mu\text{s}$ , 垂直回扫每域 20 线、水平回扫  $10.9 \mu\text{s}$ 。
  - (a)  $63.6 \mu\text{s}$  是如何得出的?
  - (b) 垂直回扫和水平回扫哪个耗时更多? 具体多耗费多长时间?
2. 欧洲的 PAL 和北美的 NTSC 哪种制式的闪烁更小? 证明之。
3. 有时, 电视信号由少于电视传输所要求的所有部分所组成。
  - (a) 室内广播电视需要多少种信号? 这些信号分别是什么?
  - (b) S-video 需要多少种信号? 这些信号分别是什么?
  - (c) 用于标准模拟电视接收的广播有多少信号? 那一类视频叫什么?
4. 在解调过程中, 如何从 NTSC 的色度信号  $C$  中提取出  $Q$  信号?
5. 一种说法是一种录像带的老的 Betamax 格式, 从 VHS 和损失来看, 是一种较好的格式。请对

这种说法给出评价。

6. 在工作站上播放 NTSC 制式的帧时一般不会出现闪烁。请问可能的原因是什么？
7. 请问数字视频信号采用色度二次采样的原因是什么？这样做为什么是可行的？
8. 普通的 TV 和 HDTV 的最显著的差别是什么？HDTV 发展的主要动力是什么？
9. 隔行扫描视频的的优点是什么？它存在哪些问题？
10. 解决隔行扫描视频所带来的问题的一种办法是去除交错处理。为什么不能通过重叠两个域来获取一幅去除交错处理的图像？请提出一些简单的去除交错处理的算法来同时保留两个域的信息。

## 5.6 参考文献

- [1] J.F. Blinn, "NTSC: Nice Technology, Super Color," *IEEE Computer Graphics and Applications*, 13(2): 17–23, 1993.
- [2] A.M. Tekalp, *Digital Video Processing*, Upper Saddle River, NJ: Prentice Hall PTR, 1995.
- [3] A. Bovik, editor, *Handbook of Image and Video Processing*, San Diego: Academic Press, 2000.
- [4] C.A. Poynton, *A Technical Introduction to Digital Video*, New York: Wiley, 1996.
- [5] J.F. Blinn, "The World of Digital Video." *IEEE Computer Graphics and Applications*, 12(5): 106–112, 1992.
- [6] C.A. Poynton, *Digital Video and HDTV Algorithms and Interfaces*, San Francisco: Morgan Kaufmann, 2003.
- [7] R. Steinmetz and K. Nahrstedt. *Multimedia: Computing, Communications and Applications*, Upper Saddle River, NJ: Prentice Hall PTR, 1995.

## 第6章 数字音频基础

音频信息对于多媒体表现非常重要,从某种角度来讲,它也最简单的一种多媒体数据。但音频信息和图像信息之间的一些显著的差异是不容忽视的。例如,在视频流中,为了加快视频的播放速度,我们可以酌情丢弃一些视频帧,但是在音频信息中,我们却不可以这么做,因为这样做会丢失整个声音信息的语义。在这一章中,我们介绍多媒体中声音的一些基本概念,在第13、14章中介绍声音压缩的一些细节问题。声音信息的数字化会涉及到信号处理中采样和量化的一些知识,所以我们在本章中介绍这些概念。

我们首先讨论声音信息的组成。然后我们讨论一下 MIDI,它用于采集、存储以及播放数字音频。接着我们讨论传输过程中涉及的音频量化的一些细节,同时简单介绍在存储和传输过程中对数字音频的处理。这里,首先讨论为什么预测编码的残差会接近0,并因此易于处理。

### 6.1 声音数字化

#### 6.1.1 什么是声音

类似于光,声音是一种波动现象,但与光不同的是,声波是一种宏观现象,一些物理器件的运动导致了空气分子的震动,进而产生声波。例如,音频系统里面的扩音器前后振动产生径向的压力波,这种波就是我们所听到的声音。(举个例子:我们把 Slinky<sup>⊙</sup>沿着它的径向强后振动,就能得到一个径向波,如果垂直于它的径向上下振动,就能得到一个横向波。)

没有空气就没有声音,例如在太空就听不到声音。因为声波是压力波,取连续值,这与取离散值的数字信号是完全不同的。因此,如果我们想要将声波作为数字信号来处理,我们首先需要将音频信号数字化。

虽然声波是径向波,它也具有一般的波的属性和行为,例如反射、折射(进入另一种具有不同密度的介质时产生的角度变化)和衍射(绕过特定的障碍物)。这些特点有助于我们制造环绕声场。

在三维空间的任意一点,声波都带有可测量的压力值,因此我们使用压力传感器将压力转换成电压差,从而通过检测某个区域的压力值来探测声音。

#### 6.1.2 数字化

图6-1显示了声音的一维性质。振动的幅值随着时间不断变化,因为压力值随着时间变化不断增加或减小[1]。幅值是连续值。我们感兴趣的是如何在计算机的存储器中处理语音数据,所以我们必须将麦克风产生的模拟信号(即连续的电压值)数字化。在处理图像数据时,我们也需要将摄像机获得的与时间相关的模拟信号数字化。数字化意味着将连续值转换成一系列的离散值(考虑到效率,主要转换成整数)。

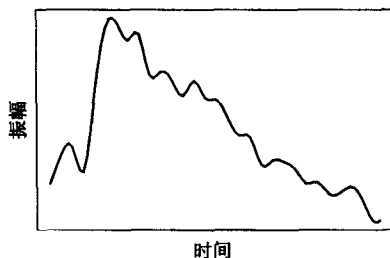


图 6-1 一段模拟信号：压力波的连续测量

⊙ Slinky 是一种弹簧玩具。——译者注

图 6-1 是二维的, 要将所示的信号全部数字化, 我们就要在每一维(时间维和幅值)上采样。采样指在一个空间中按照一定的取值间隔对我们感兴趣的一个量进行测量。按照确定的时间间隔在时间轴上进行的采样简称“采样”, 采样的速度称为采样率。图 6-2a 显示的就是这种类型的数字化。

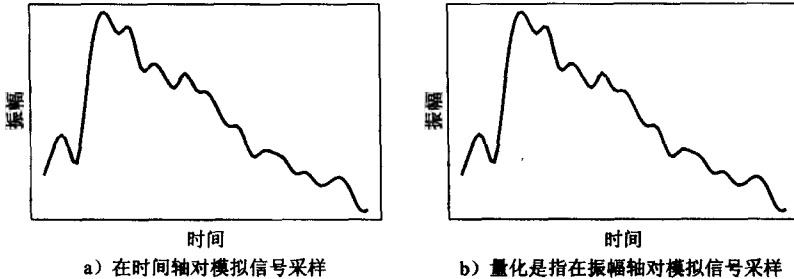


图 6-2 采样和量化

对于音频信号, 采样率一般从 8kHz (每秒钟 8000 个采样点) 到 48kHz。人能够听见从 20Hz~20kHz 的声音。高于 20kHz 的声音称为超声。人发出的声音最高能够达到 4kHz, 我们的采样率的下限最少应该是这个频率的 2 倍 (参见后面对奈奎斯特采样率的讨论)。现在我们能够达到的采样率大概是 8~40kHz。

如图 6-2b 所示, 在振幅维 (或者说电压轴) 的采样称为量化 (quantization)。虽然我们只讨论等采样间隔下的均匀采样, 但非均匀采样也是可能的。非均匀采样一般不用在时间维, 但是可以用于量化 (参见下文的  $\mu$ -law 法则)。典型的均匀量化率是 8 位和 16 位。8 位量化把纵轴分成 256 个区间, 16 位量化把纵轴分成 65 536 个区间。

要决定怎样把音频数字化, 我们需要回答以下一些问题:

- 1) 什么是采样率?
- 2) 怎样精细地进行数据量化? 量化是均匀的吗?
- 3) 音频数据是怎样的格式? (即是什么样的文件格式?)

### 6.1.3 奈奎斯特理论

如果使用了足够的正弦函数, 我们可以把信号分解成一系列正弦函数的和。图 6-3 显示了经过加权的一些正弦函数是如何叠加成复杂的信号的。这里频率是绝对的衡量标准, 音高则是一种感觉上的、主观上的声音质量——一般而言, 音高是相对的。音高和频率是按如下方式联系在一起的: 将音符 A 以正好 440Hz 作用在中央 C 上。这个音符上的一个八度音程与两倍的频率相关并引导我们到另一个 A 音符。因此, 在钢琴的中央 A (“A4” 或 “A440”) 设为 440Hz, 下一个 A 就是 880Hz, 高了一个八度音程。

在这里, 我们把和声定义成频率为基音整数倍的一系列的乐音。图 6-3 就包含一些和声。

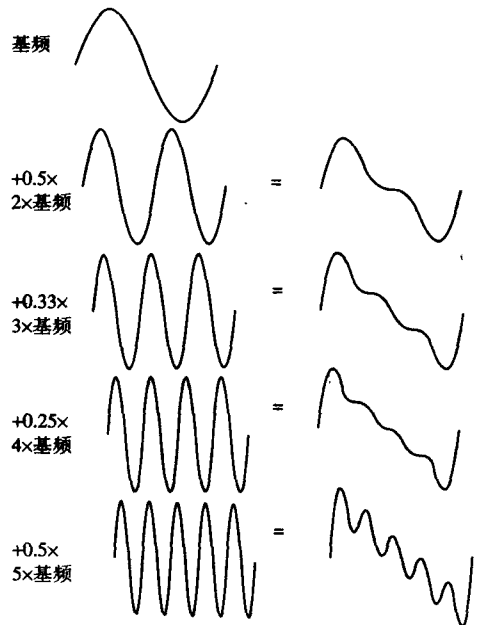


图 6-3 用正弦信号叠出复杂的信号



如果我们允许基音频率的非整数倍语音,如允许非 A 音符,就产生了一段复杂的声音。无论如何,每段声音都是正弦波叠加的结果。图 6-4a 显示了一个正弦函数,它是单一的、单纯的并具有周期性(只有电子乐器能够产生如此单调的声音)。

如图 6-4b 所示,如果采样频率和语音的真实频率一致,我们会检测到一个错误的信号,它仅仅是一个常数,频率为 0。如果使用语音频率的 1.5 倍频率来采样,如图 6-4c 所示,我们会得到一个比真实频率小的假频(alias),该频率只有真实频率的一半(采样信号的波长,即波峰到波峰间的距离是实际信号的 2 倍)。在计算机制图中,人们做了很多的工作,通过各种抗锯齿的方法来消除这种假频现象。假频是指不属于原始信号的信号。基于以上原因,为了得到正确的采样,我们需要使用的采样频率至少是信号中最高频率的两倍。这就是奈奎斯特采样率。

奈奎斯特是贝尔实验室的一位著名的数学家,奈奎斯特定理就是以他的名字命名的。更一般地,对于一个限带信号(band-limited signal,即信号的频率分量的下界为  $f_1$ ,频率上界为  $f_2$ ),那么采样频率至少是  $2(f_2 - f_1)$ 。

假设我们有一个固定的采样率,因为不可能恢复出高于采样频率一半的频率,很多的系统都带有抗锯齿的滤波器(filter),用来限制采样器的输入频率不得低于  $1/2$  的采样频率。容易混淆的是,习惯上人们把奈奎斯特采样频率的一半称作奈奎斯特频率。因此,对于我们固定的采样率而言,奈奎斯特频率就是该采样频率的一半。信号中最高可能的频率成分与频率相同。

注意,信号的真实频率和它的假频对称分布在频率轴上,在与所使用的采样频率有关的奈奎斯特频率两侧。因此,采样频率所对应的奈奎斯特频率通常称为“折线”频率。也就是说,如果采样频率小于实际信号频率的两倍,但是大于实际信号频率,则假频等于采样频率减去真实频率。例如,真实频率是 5.5kHz,采样频率是 8kHz,那么假频为 2.5kHz:

$$f_{\text{假频}} = f_{\text{采样频率}} - f_{\text{真实频率}}, f_{\text{真实频率}} < f_{\text{采样频率}} < 2 \times f_{\text{真实频率}} \quad (6.1)$$

同时,任何一个频率的两倍都可以作为原频率的采样点。事实上,将真实频率加减采样频率的整数倍后,就得到一个可能的假频。而用采样频率采样时,这样的假频能给出同样一组采样。

因此,再次强调,如果采样频率不仅小于真实频率的两倍,而且小于真实频率,那么假频等于几倍的采样频率减去真实频率,其中  $n$  是大于真实频率的  $n$  倍于采样频率的最小整数,我们求采样频率的倍频中比真实频率大的最小频率,用它减去真实频率,得到的差就是假频。例如,在真实频率介于 1.0 和 1.5 倍的采样频率之间时,假频就等于真实频率减去采样频率。

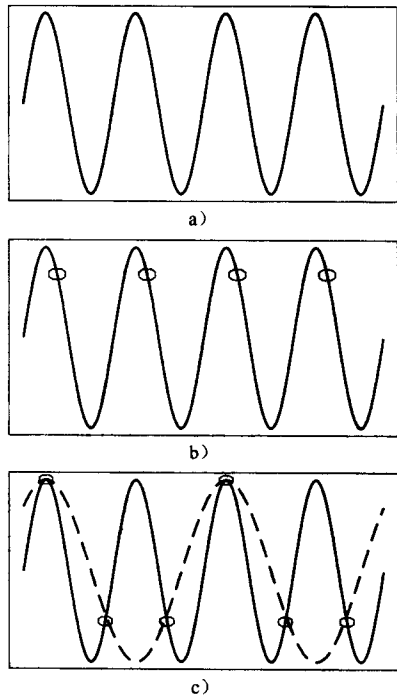


图 6-4 假频

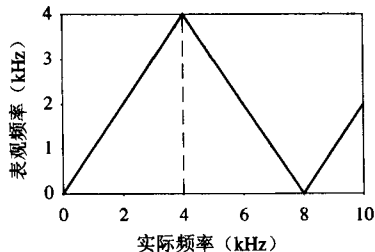


图 6-5 用 8000Hz 采样产生的正弦频率的折叠效应,折叠频率是 4000Hz,用虚线表示

一般来讲, 一组拥有相同采样点的正弦信号的表现频率是该组信号中最低的频率。图 6-5 表明了输入(真实)频率的表现频率的关系。

#### 6.1.4 信噪比

在模拟系统中, 随机的波动会在信号里产生噪声(noise), 测量电压会因此而变得不准确。正确信号的能量和噪声能量的比称为信噪比(Signal-to-Noise Ratio, SNR)。信噪比是信号质量的衡量标准之一。

信噪比的常用单位是分贝(dB), 1dB 等于 1/10 贝尔。信噪比是用电压平方比取 10 为底的对数来定义的:

$$SNR = 10 \log_{10} \frac{V_{\text{信号}}^2}{V_{\text{噪声}}^2} = 20 \log_{10} \frac{V_{\text{信号}}}{V_{\text{噪声}}} \quad (6.2)$$

信号的能量和电压的平方成正比。例如, 如果信号的电压是噪声电压的 10 倍, 那么信噪比  $SNR = 20 \log_{10}(10) = 20 \text{dB}$ 。

从能量的角度说, 如果 10 把小提琴演奏所产生的能量是 1 把小提琴演奏所产生能量的 10 倍, 那么用 dB 作单位, 就得到能量比是 10dB, 或者说是 1bel。注意, dB 始终是用比例来定义的。在日常环境中描述声音强度, 使用我们刚刚可以听到的 1kHz 的声音作分母。我们把周围环境中的声音和我们能够听到的最轻的声音求能量比, 然后得到 dB 值, 用它来衡量声音的强度。表 6-1 给出了不同声音的近似强度。

表 6-1 常见声音的强度 (dB)

听力阈值	0
树叶摇动	10
很安静的房间	20
一般房间	40
交谈	60
繁华街道	70
吵闹的收音机	80
火车穿过车站	90
打铆机	100
不舒服的阈值	120
痛苦的阈值	140
伤及鼓膜	160

#### 6.1.5 信号量化噪声比

在数字信号中, 我们存储是离散值。对于一个数字音频信号来说, 每个采样的精度取决于每个采样的位数, 通常是 8 位或者 16 位。

除了原始的模拟信号中存在的噪声外, 离散化会带来其他一些误差。比如, 若电压值的范围是 0~1, 但是我们只能 8 位来存储, 所以只能将原本连续的电压值离散成 256 个不同的电压值。取整的过程必然会带来误差。尽管这不是真正的噪声, 但还是称之为量化噪声(或量化误差)。称之为噪声是因为这种错误是在不同的采样点之间随机出现的。

量化质量使用信号量化噪声比(SQNR)来描述。量化噪声是指某个采样时间点的模拟值和最近的量化值之间的差。误差最大可以达到离散间距的一半。

如果每个采样点的量化精度是  $N$  位, 数字信号的取值范围是  $-2^{N-1}$  到  $2^{N-1}-1$ 。因此, 如果实际的模拟信号的范围是  $-V_{\max} \sim +V_{\max}$ , 那么每个量化级代表的电压是  $2V_{\max}/2^N$  或者  $V_{\max}/2^{N-1}$ 。信号量化噪声比可以只考虑峰值情况, 取  $V_{\text{signal}}$  为  $2^{N-1}$ , 同时把分母  $V_{\text{quan\_noise}}$  取 1/2。上面两者的比值就是信号量化噪声比的简单的定义<sup>①</sup>:

$$\begin{aligned} SQNR &= 20 \log_{10} \frac{V_{\text{signal}}}{V_{\text{quan\_noise}}} = 20 \log_{10} \frac{2^{N-1}}{\frac{1}{2}} \\ &= 20 \times N \times \log 2 = 6.02N(\text{dB}) \end{aligned} \quad (6.3)$$

① 这个比率实际上是峰值信号-量化-噪声比值(PSQNR)。

换句话说, 采样点中每个位增加 6dB 的分辨率, 因此 16 位能够达到的  $SQNR$  为 96dB。

刚才我们研究的是最坏的情况。另一方面, 如果我们假设输入信号是正弦信号, 那么量化误差在统计上是无关的, 在  $0 \sim 1/2$  量化间隔之间呈正态分布。可以证明,  $SQNR$  的表达式 ([2] 的 37 页) 是:

$$SQNR = 6.02N + 1.76(\text{dB}) \quad (6.4)$$

从公式中可以看出,  $N$  越大,  $SQNR$  越大。这说明对模拟信号的逼近越精确, 系统能够提供的音质越好。

一般而言, 数字音频采样的精度是每个采样 8 位 (相当于电话的音质) 或每个采样 16 位 (相当于 CD 的音质)。实际上, 用 12 位左右就能很好地再现原声音了。

### 6.1.6 线性量化和非线性量化

前面提到过, 采样通常存储为均匀分布的离散值, 我们称之为线性格式 (linear format)。但因为可用的位数有限, 我们自然会想到应该更加关注人的听觉范围内声音, 所以可以设置一套非线性量化体制使得人有更好的听觉享受。

我们量化的是声音的幅度, 即信号的声音大小。在第 4 章我们讨论了韦伯定律, 它指出声音的一个特性, 即声音自身的强度越大, 就越需要更大的振幅来让我们感受到声音的变化。一般来说, 韦伯定律指出, 要产生同样的感知所需要的增幅是和原来的绝对值是成比例的:

$$\Delta \text{Response} \propto \Delta \text{Stimulus} / \text{Stimulus} \quad (6.5)$$

举个例子, 如果我们能够感受到从 10 磅到 11 磅的变化, 那么从 20 磅开始, 需要增加到 22 磅, 我们才能感受到同样的重量变化。

加入一个固定系数  $k$ , 我们就有微分方程:

$$dr = k(1/s)ds \quad (6.6)$$

其中  $s$  是激励,  $r$  是响应值。积分后, 我们求得解为:

$$r = k \ln s + C \quad (6.7)$$

其中  $C$  为积分常数。解也可以写成:

$$r = k \ln (s/s_0) \quad (6.8)$$

其中  $s_0$  是产生响应的最小激励值 (当  $s = s_0$  时,  $r = 0$ )。

利用这一感知特性的非均匀量化方案使用了对数。因为根据式 (6.8) 绘制的对数曲线显示, 如果我们沿着  $s$  轴均匀增加  $s$  值, 在  $r$  轴得到的不是均匀增加的响应值。

我们希望在  $r$  轴获得均匀的增加值。因此在非线性量化中, 首先把模拟信号值由原始的  $s$  空间映射到  $r$  空间, 然后再均匀量化  $r$  空间的值。这样, 在信号值的低端,  $r$  轴上的一个间隔在  $s$  轴能覆盖很大的范围, 在信号值的高端,  $s$  轴的一个间隔能够覆盖  $r$  轴上很大范围。

这个用于音频的定律称为  $\mu$  律 ( $\mu$ -law) 编码 (为了书写方便, 我们也称之为  $\mu$  律)。欧洲的电话系统中也使用了相似的技术, 称为  $A$  律 ( $A$ -law)。

这些编码算法的方程如下:

$\mu$  律:

$$r = \frac{\text{sgn}(s)}{\ln(1+\mu)} \ln \left\{ 1 + \mu \left| \frac{s}{s_p} \right| \right\}, \quad \left| \frac{s}{s_p} \right| \leq 1 \quad (6.9)$$

$A$  律:

$$r = \begin{cases} \frac{A}{1 + \ln A} \left( \frac{s}{s_p} \right), & \left| \frac{s}{s_p} \right| \leq \frac{1}{A} \\ \frac{\operatorname{sgn}(s)}{1 + \ln A} \left[ 1 + \ln A \left| \frac{s}{s_p} \right| \right], & \frac{1}{A} \leq \left| \frac{s}{s_p} \right| \leq 1 \end{cases} \quad (6.10)$$

$$\text{其中 } \operatorname{sgn}(s) = \begin{cases} 1 & s > 0 \\ -1 & \text{其他} \end{cases}$$

图 6-6 给出了两个函数的曲线。 $\mu$ 律编码器中一般设  $\mu = 100$  或者  $\mu = 255$ ,  $A$  律编码器中一般设  $A = 87.6$ 。

公式中的  $s_p$  是信号的峰值,  $s$  是信号的当前值, 也就是说  $s/s_p$  的值域是  $-1 \sim 1$ 。

上述定律的主要思想是: 首先将  $s/s_p$  转换成  $r$  值, 再将  $r$  值均匀量化, 然后就可以存储或者传输信号。由于人的感知不一致性, 这样就可以将大部分的位用于存储信号中的改变容易被人感知的信息。

为了说明这个问题, 考虑在图 6-6 中,  $|s/s_p|$  在 1.0 附近发生了小的变化, 与此对应的曲线非常平坦。显然,  $s$  值在平坦区域产生变化要远大于在原点附近量化后  $r$  值产生的变化。也正是在很安静的环境中, 我们能够感受到声音的细微变化。 $\mu$ 律关注的正是这个范围的声音变化。

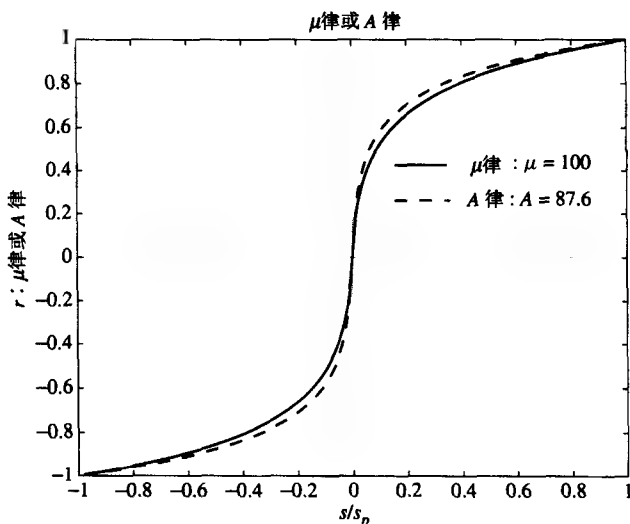


图 6-6 音频信号的非线性变换

编码时首先作  $\mu$  律转换, 然后对转换结果作均匀量化, 产生的结果对于输入来说是非线性转换。其中的对数函数对于振幅小的声音更加敏感。如果我们用固定数目的位来编码声音, 音量小的声音产生的量化误差会小于音量高的声音。因此  $\mu$  律使得信噪比在输入信号范围内分布更加均匀。

这项技术是基于人类的感知的, 也是最简单的“感知编码”。有趣的是, 一份有说服力的统计表明, 人类喜欢的音域也主要是在低音区域。所以我们应该把更多的位用在声音最频繁出现的区域, 那里的概率密度最高。所以这种编码技术也是基于统计结果的。

总的说来, 首先对模拟信号做一个对数转换 (电信业的说法为“压缩”), 然后进行采样, 再将采样结果数字化 (通过一个模/数转换器)。输入信号的振幅越大, 压缩率越高。模/数转换器对“压缩”后的信号作均匀量化。编码完毕后, 如果我们要播放声音, 那么就需要模拟信号, 此时

就要将编码后的值转换回去，首先使用数/模转换器，得到模拟信号后，通过一个“扩展”电路作反对数转换。这个转换过程称为压缩扩展。如今，压缩扩展技术在数字处理领域仍然在使用。

音频处理中的 $\mu$ 律用来对声音进行非均匀的量化。总的说来，我们愿意把更多的位用在人们感觉最灵敏的声音区域。在理想的情况下，应该根据人对不同刺激信号所做出的响应曲线来划分位。这样，我们可以把位划分给那些用较小的激励就能产生较大响应的信号段。

压缩扩展技术体现了给信号划分位的一种思想：尽量使得结果能够获得更好的分辨率。这一点和以前的均匀量化的思想是不同的，体现了非均匀量化的思想。 $\mu$ 律（或者 A 律）编码就是这种想法的一个应用。

6.1.7 音频滤波

在进行采样和模/数转换前，常常需要对音频信号作滤波以消除不需要的频率。保留哪些频率要取决于具体应用。对于语音信号，一般都保留 50Hz~10kHz 的频率。其他频率成分通过带通滤波器（band-pass filter）过滤掉（带通滤波器能够筛选出高端信号和低端信号，也称为带限滤波器）。

一段音乐信号通常是从 20Hz~20kHz（20Hz 是由烦躁的大象发出的低沉的声音，20kHz 是我们能听到的最大尖叫声），所以音乐的带通滤波器将筛掉除此之外的频率。

在一段信号中，有时候尽管我们已经将类似噪声的高频信号过滤掉了，在经过数/模转换后，那些高频信号会再次出现。这是因为我们在采样和量化时，将原来的连续平滑的信号转换成阶梯函数。从理论上讲，这种离散的信号包含各种可能的频率成分。因此，在解码器端，经过数/模转换后需要使用一个低通滤波器（其作用和在编码器端过滤高频信号的带通滤波器一样）。

这里我们还是回避了一个问题：在语音或者音频应用中，每个采样点需要多少位。本章的练习中会涉及这个问题。

常用的音频文件格式包括 AU（用于 UNIX 工作站）、AIFF（用于 MAC 和 SGI 机）、WAV（用于 PC 和 DEC 工作站）。在第 14 章会讨论 MP3 压缩文件格式。

6.1.8 音频质量与数据率

如果在量化过程中使用的位越多，那么未压缩数据的数据率就会越大。立体声信息所需的带宽是普通的单声道信息所需带宽的两倍。表 6-2 显示了音频质量和数据率以及带宽之间的关系。

在模拟信号系统中，术语带宽（bandwidth）指一个器件的响应能力或者传输能力。如果用  $x$  轴表示频率， $y$  轴表示传输函数，那么带宽函数一般都是常数函数。半能量带宽（Half-power Bandwidth, HPBW）指含有最大能量一半的一段频率范围。因为  $10\log_{10}(0.5) \approx -3.0$ ，所以也用 -3dB 带宽来表示 HPBW。

所以在模拟设备中，带宽使用的是频率的单位赫兹（Hz），赫兹的物理意义是指每秒的周期数。对数字设备，在某个固定带宽下能传输的数据量用每秒通过的位（bps）或者每段时间的字节数来表示。不管是模拟设备还是数字设备，带宽都是指单位时间内通过的数据量。

表 6-2 音频采样系统中的数据率和带宽

质量	采样率 (kHz)	每个采样的位数	单声道/立体声	数据率 (非压缩) (kB/sec)	频带 (Hz)
电话	8	8	单声道	8	200~3 400
AM 广播	11.025	8	单声道	11.0	100~5 500
FM 广播	22.05	16	立体声	88.2	20~11 000
CD	44.1	16	立体声	176.4	5~20 000
DAT	48	16	立体声	192.0	5~20 000
DVD 音频	192 (最大)	24 (最大)	最高为 6 信道	1200.0 (最大)	0~96 000 (最大)

电话学中使用 $\mu$ 律编码（在欧洲使用A律编码）。其他的格式使用线性量化。使用式（6.9）中的 $\mu$ 律编码时，可以把数字信号的码字长度从8位扩展到12位或13位。

有时候我们需要将表6-2中的数据率转换成以“字节/分”为单位。例如，未经压缩的CD品质的立体声数字音频信号的数据率是10.6 MB/分，取整可以写成10MB/分。

### 6.1.9 合成的声音

数字化的声音首先要转换成模拟信号，然后才能播放。对于存储的采样音频，有两种不同的处理方法。第一种是调频（Frequency Modulation, FM），第二种是波形表法（Wave Table）。

在第一种方法中，我们通过在一个载波正弦信号中加入一个涉及调频信号的项使原来的正弦信号发生改变。如果我们使用余弦函数来载波，在余弦函数的参数中加入第二个余弦函数，这样就是在求一个余弦值的余弦，结果会得到一个合成后的声音。用一个随时间变化的振幅函数来增强整个信号，用另外一个随时间变化的函数来增强参数中的余弦信号，从而构成和声。最后在每个余弦函数的参数中加入一个常数，这样就得到了一个非常复杂的函数。

例如，图6-7a显示了函数 $\cos(2\pi t)$ 的图像，图6-7b是 $\cos(4\pi t)$ 的图像。图6-7c比较有趣，它显示了一个余弦函数的余弦的图像。图6-7d中显示了一段调频信号，载波频率是2，调制频率是4。很显然，如果我们考察一个更加复杂的信号，例如下面的公式[3]所描述的信号：

$$x(t) = A(t) \cos[\omega_c t + I(t) \cos(\omega_m t + \phi_m) + \phi_c] \quad (6.11)$$

我们可以合成任何复杂的信号。

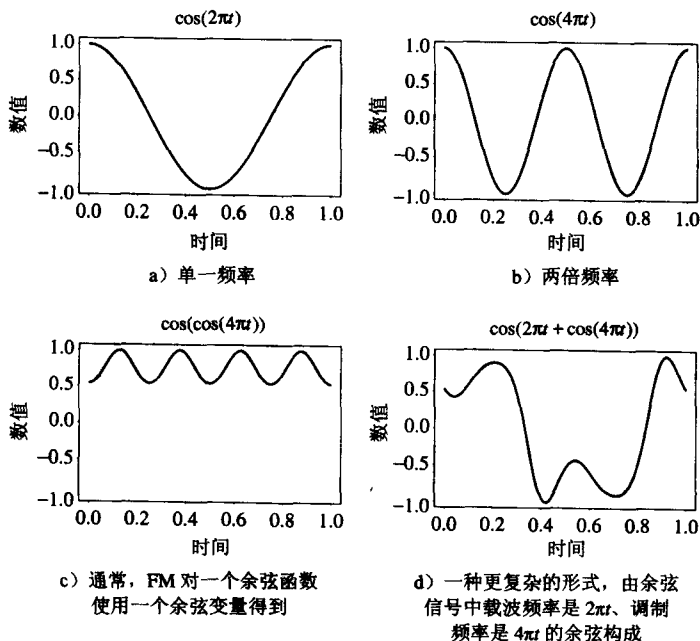


图 6-7 调频

利用上面这个调频方程，我们生成了一个信号，它的基本载波频率是 $\omega_c$ ，调制频率是 $\omega_m$ 。在图6-7d中， $\omega_c = 2$ ， $\omega_m = 4$ 。相位常数 $\phi_m$ 以及 $\phi_c$ 会产生一些时间偏移，从而生成更加有趣的声音。公式中和时间相关的函数 $A(t)$ 称为包络，它描述声音在各个时间段的峰值，可以用来产生声音的渐进增强或者渐进减弱的效果。例如，吉他的琴弦有激发阶段（attack period）、衰减阶段

(decay period)、持续阶段(sustain period)和释放阶段(release period)。

参数中和时间相关的函数  $I(t)$  用来改变声音中的调制频率,使听者产生和声的感觉。如果  $I(t)$  比较小,我们主要听到的是低频的声音。当  $I(t)$  较大的时候,我们就能够听到较高频的声音。在常用的 Creative Labs Sound Blaster PC 声卡的低端版本中使用了调频技术。

由数字信号合成声音的一种更精确的方式是波形表合成(wave-table synthesis)。在波形表合成中,数字采样存储的是来自真实乐器的声音。因为波形表存储在声卡的存储器中,可以通过软件来管理,所以可以对声音进行混音、编辑和增强等处理。在声音再现方面波形表比调频有着更好的效果。为了节约内存空间,还可以应用一些专门的技术,比如采样轮循(sample looping)、移调(pitch shifting)、数学插补(mathematical interpolation)以及多项式数字滤波(polyphonic digital filtering) [4, 5]。

例如,如果一首歌的音调对于你的嗓音来说太高了,那么改变这首歌的音调会非常有用。可以通过数学方法将波形表作一个移位,从而产生较低的声音。不过这种扩展应用也只能应用在这些领域,否则就会产生声音误差。波形表中有时存储了乐器在不同音调上的采样值,所以改变音调不会产生很大的影响。因为数据的存储量更大,所以使用波形表比使用调频开销更大。

## 6.2 MIDI: 乐器数字化接口

波形表文件提供了一种准确再现真实乐器声音的方法,但是文件非常大。对于一些简单的音乐来说,一些调频合成器生成的音频信号就能满足我们的要求,通过声卡可以很容易就生成这些声音。在 PC 主板的扩展槽上增加一块声卡,它能够通过连接在主板上的扩音器处理输出声音,通过连接到计算机上的麦克风来录制声音,还能处理存储在磁盘上的音频文件。

如果在一些多媒体项目中,我们想对一些声音使用声卡的默认处理,那么我们可以使用一种简单的脚本语言以及硬件设置方案——MIDI。

### 6.2.1 MIDI 概述

MIDI 是 Musical Instrument Digital Interface 的缩写,它诞生于 20 世纪 80 年代。它规定了一套被电子音乐界广泛采用的协议,该协议使得计算机、合成器、键盘以及其他一些音乐器件能够互相交互。合成器用来产生合成音乐,它被集成在声卡上,采用前文介绍的两种合成方法中的一种。绝大多数的合成器都支持 MIDI 标准,所以在一个合成器上生成的音频文件能在另一个合成器上播放,而且播放效果非常相近。计算机需要配有专门的 MIDI 接口,大部分声卡都集成了这一接口。声卡还必须配有数/模转换器以及模/数转换器。

MIDI 是一种脚本语言,它对代表某种声音产品的“事件”编码。MIDI 文件一般都非常小。举个例子,一个 MIDI 事件可能会包含一个音符的音调、延时和音量等数据。

#### 1. 术语

合成器(synthesizer)过去(现在也应该是)指一个独立的声音生成器。它可以改变音调、音量、音色(音调就是乐器演奏的音高——例如 C 调, G 调)。它还可以改变其他的声音特性,例如激发时间和延续时间。一个专业的合成器(音乐家使用的)配有微处理器、键盘、控制面板和内存等。不过现在的 PC 机的声卡中都集成了比较廉价的合成器,产生声音的模块一般被称为声音模块。

音序器(sequencer)最初是指一种用来以 MIDI 数据形式存储和编辑一系列音乐事件的专用硬件。现在多指计算机上用于编辑音乐的软件。

MIDI 键盘(keyboard)不会产生声音,而是产生 MIDI 指令序列,这些指令系列称为 MIDI 消息。这有点像汇编代码,通常只包含很少的字节。你有时只需要 3kB 的空间就能存储 3 分钟的

139 音乐。相比之下，一个波形表文件（WAV）需要 10MB 空间来存储 1 分钟的音乐。在 MIDI 中，人们习惯把键盘称为键盘控制器。

## 2. MIDI 的常用概念

在音序器中，音乐是按照音轨来组织的。在录音和回放的时候，可以打开或者关闭一个音轨。通常，某种乐器和一个专门的 MIDI 通道相对应。MIDI 通道是用来分隔消息的。一共有 16 个通道，按照 0~15 编号。消息的后 4 位（重要性最低的位）用来存储通道编号。每个通道和某种乐器相对应，例如，通道 1 代表钢琴，通道 10 代表鼓。如果需要，可以在中途更换乐器，也可以用其他的乐器和特定的通道相对应。

在消息中，通道还可以用作占位符。如果一个消息的前 4 位全部是 1，那么这个消息就被认为是系统通用消息。

除了通道消息（含有通道编号）之外，还有其他的一些消息需要发送，例如通知所有的乐器更改基调以及节拍的消息。这些消息称为系统消息。还可以给某个乐器通道发送特定消息，使得该通道能发出一些事先没有指定的声音。后面我们将会详细介绍这些消息。

合成乐器响应 MIDI 消息的方式通常是将不属于当前通道的“播放声音”消息忽略。如果同时有几个属于当前通道的消息，就好像在钢琴上同时奏响几个音符，那么乐器只要是多声部的，它就会响应。也就是说，乐器会同时演奏几个音符。

术语声部（voice）和音色（timbre）很容易混淆。后者在 MIDI 中是一个专门术语，特指我们想要模拟的乐器，例如钢琴、小提琴。它指不同乐器各自的声音特点，如果一个乐器（或者说一个声卡）是多音色的，那么它能够同时播放几种不同乐器的声音（例如，钢琴、贝司、鼓）。

另一方面，“声部”有时候也被音乐家用来表示音色的意思，但在 MIDI 中，声部指一个声音模块同时能播放的不同的音色和音调。一个合成器能拥有许多种声部（一般有 16、32、64、256 种）。每种声部能够独立工作且能同时生成不同音色和音调的声音。

术语复音（polyphony）指能同时发生出的声部的数目。一个典型的语音模块应该能够产生有 64 声部复音。（指同时能够发出 64 种不同音符），同时又是“16 部多音色”（使声音有 16 种乐器同时演奏的效果）。

不同的音色是通过编配程序（patch）来生成的。编配程序是一系列的控制指令，通过它来定义一个专门的音色。编配程序一般存储在数据库里，称之为“音色库”。一些真正的发烧友拥有专门的编配程序编辑软件。

关于什么乐器和什么通道对应，有一个标准的映射机制，这个机制获得大家的一致支持，称之为通用 MIDI（General MIDI）。在通用 MIDI 中，有 128 个编配程序和标准乐器相对应，第 10 通道是为打击乐器预留的。

在绝大多数乐器中，一个典型的消息是 Note On（例如，按下了一个键），其中包含通道编号、音调、强度（即音量）。对于打击乐器，音高数据指用哪一种鼓。因此一个 Note On 消息包含一个状态字节（什么通道，什么音调），后面是两个数据字节。Note On 消息后一般跟随一个 Note Off 消息（释放一个键），该消息也有特定的音调（关闭哪一个音符），为了保持一致性，还包含强度数据（一般都设成 0，或者被忽略）。

140 MIDI 状态字节中的数据的取值从 128~255，每个字节的取值从 0~127。实际使用中，每个 MIDI 字节是 8 位，另外还有一个开始位，一个结束位，这两个位都被设为 0。所以 MIDI 中每个字节有 10 位。图 6-8 显示了一个 MIDI 数据流。

很多的 MIDI 设备都是可编程的，这意味着它们带有特定的过滤器能够改变高音、低音效果，以及改变描述随时间改变的声音幅值包络线。图 6-9 显示了一个乐器对于 Note On/Note Off 消息的响应的模型。



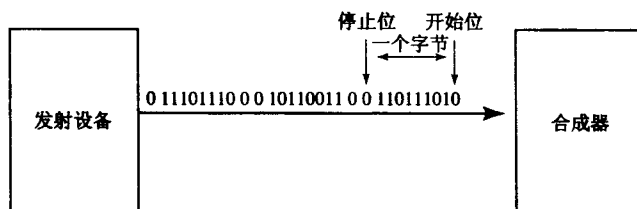


图 6-8 10 位字节数据流。在典型的 MIDI 消息中，流中的信息是 {状态字节, 数据字节, 数据字节} = {Note On, Note Number, Note Velocity}

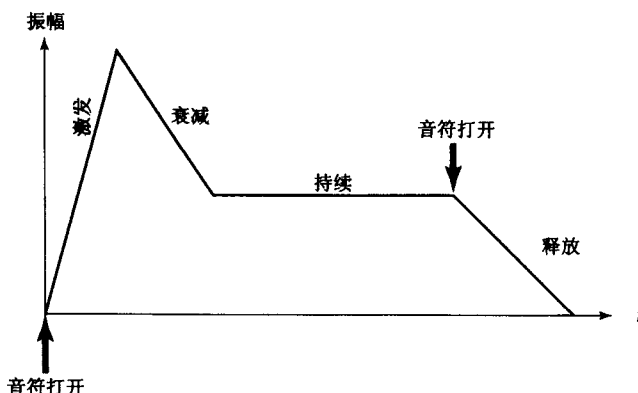


图 6-9 一个音符的振幅-时间图

MIDI 音序器（编辑器）允许你使用标准音符来编辑，如果需要，也可以直接编辑数据。MIDI 文件可以存储波形数据。WAV 文件的特点是它可以更加精确地存储乐器的声音。采样器用于采样音频数据，例如，“电子鼓”专门用来存储真实鼓的 WAV 数据。

141

音序器中使用了一些专门技术来利用已有的音乐产生更多的音乐。例如，可以在一段音乐的几个小节间循环。音量随着时间变得更加容易控制，这被称为时变振幅调制。更有趣的是，音序器也可以实现时间扩展或者压缩而不会使音高产生变化。

因为可以改变一个采样乐器的音调，所以如果音调改变很大，那么得到的声音就会变得刺耳。为了解决这个问题，采样器中使用了多采样机制。一个声音在被录下的时候经过多个带通滤波器，记录结果被分配给多个按键。这样使得音调变化带来的频率变换更加可信，因为每个音符发生了很小的变化。

## 6.2.2 MIDI 硬件

MIDI 硬件中含有一个 31.25kbps 的串口连接，其中使用了 10 位的字节，每个字节包含一个开始位和结束位。一般而言，MIDI 器件可能是输入器件，或者是输出器件，不可能两者皆是。

图 6-10 是一个常用的合成器。调制旋钮用来增加颤音。音调旋钮用来调节频率，有点类似轻轻地按在吉他的琴弦上。还有一些其他的控制装置，例如脚踏板、滑动条等。

物理的 MIDI 端口配有分别标记为 IN 端口、OUT 端口的 5 针连接器以及第三个连接器 THRU。最后一个数据通道只是复制进入 IN 通道中的数据。MIDI 通信是半双工的。器件通过 MIDI IN 连接器件用于输入所有的 MIDI 数据，MIDI OUT 连接器用于输出它产生的 MIDI 数据。设备通过 MIDI THRU 连接器响应它从 MIDI IN 中接收到所有数据（仅仅响应这些数据，器件自己产生的所有数据都通过 MIDI OUT 输出了）。这些端口都在声卡上或者在一个接口上，可能是在 PC 扩展

槽上的单独声卡上，也可能使用一个专门的连接串口或者并口的接口。

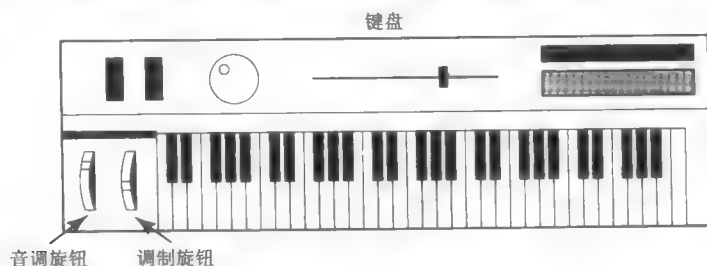


图 6-10 MIDI 合成器

图 6-11 显示了一个典型的 MIDI 音序器设置。键盘的 MIDI OUT 连接到合成器的 MIDI IN 端口，然后将该合成器的 THRU 连接到一些其他的声音模块上。在录音过程中，一台带有键盘的合成器给音序器发送 MIDI 消息，音序器记录这些消息。在播放过程中，消息从音序器发送到各个发声模块和合成器，最后通过合成器来播放音乐。

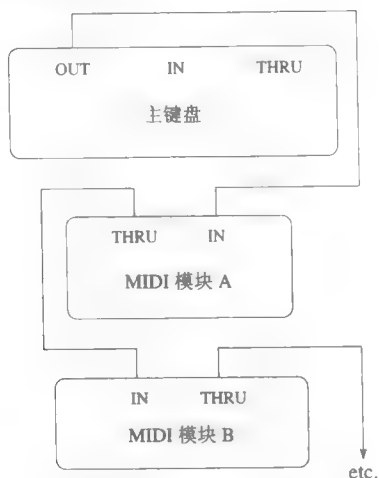


图 6-11 典型的 MIDI 设置

### 6.2.3 MIDI 消息的结构

MIDI 消息可以被分成两种，如图 6-12 所示，一种是通道消息，另一种是系统消息，还可以对这两种消息进一步划分。接下来，我们将会对每一种消息进行解释。

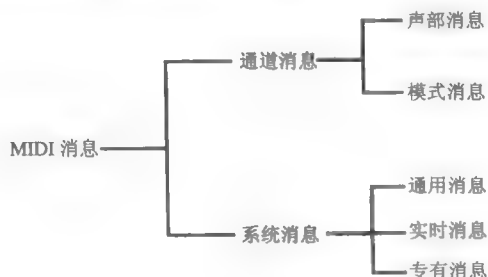


图 6-12 MIDI 消息分类

表 6-3 MIDI 声部消息

声部消息	状态字节	数据字节 1	数据字节 2
Note Off	&H8n	键号	Note Off 速度
Note On	&H9n	键号	Note On 速度
Polyphonic Key Pressure	&HAn	键数	数量
Control Change	&HBn	控制器号	控制器值
Program Change	&HCn	程序号	空
Channel Pressure	&HDn	压力值	空
Pitch Bend	&HEn	MSB	LSB

注：&H 表示 16 进制，状态字节中的  $n$  是 16 进制值，表示一个通道号。所有的值都是 0 到 127，除了控制器号是 0 到 120。

### 1. 通道消息

一个通道消息最多 3 个字节。第 1 个字节是状态字节（操作码），其中的最高位设置为 1。四个低位标记这个消息属于 16 个通道中的哪一个，剩下的 3 个位用来存储信息。对于数据字节，最高位可以设置成 0。

#### （1）声部消息

这种通道消息用来控制一个声部（即发布一个音符播放或停止的消息），同时将一个按键事件编码。声部消息还可以用来指定控制效果，比如延时、抖音、颤音以及音调变化。表 6-3 列举了这些选项。

对于 Note On 和 Note Off 消息来说，速度（velocity）表示按键的速度。一般来讲，对于一个高速的按键，合成器应该产生更加响亮和清脆的声音。Note On 消息使得一个音符出现，合成器要尽量使得这个音符尽量听起来像真实的乐器演奏效果。Pressure 消息可以用来在播放过程中改变音调。Channel Pressure 消息是一个通道（乐器）对应的按键的压力值，在该通道上播放的音符都会有同样的效果。其他的压力消息，比如 Polyphonic Key Pressure（也称为 Key Pressure 消息）设定若是按下多个键，应该发出多大的音量，每个音符应该拥有怎样的音量才能产生好的和弦。压力也称作触后（aftertouch）。

Control Change 指令用于设置各种不同的控制器（例如，衰减、颤音等）。不同的产品通过不同的指令来实现他们各自的功能。其中，1 号控制器的功能类似于调制按钮（用于颤音）。

例如，一个 Note On 消息后有两个字节，一个字节用来设定音符，另一个字节用来设定速度。因此，要在 13 号通道上以最高速度来播放 80 号音符，MIDI 设备就会发送下面三个 16 进制值：&H9C&H50&H7F（16 进制数字范围是 0~15，因为它用来标记 1~16 号通道，“&HC”表示通道 13）。音符被编号，如中音 C 是 60。

为了（有效地）同时播放两个音符，首先我们要发送一个 Program Change 消息给每一个通道。Program Change 指在一个特定的通道上转载特定的编配程序。这样我们就在两个不同的通道上附加不同的音色。然后顺序发送两个 Note On 消息，把两个通道都打开。或者，我们可以先给一个通道发送 Note On 消息，然后在发送 Note Off 消息给这个通道之前，发送另一个 Note On 消息给该通道（此时用另一个音高）。这样我们就能在一件乐器同时播放出不同的音符了。

Polyphonic Pressure 指在多个乐器上同步演奏不同的音符时，每个音符间的强度。Channel Pressure 指在一个乐器上演奏一个音符时设定的强度值。

#### （2）通道模式消息

通道模式消息是一种特殊的 Control Change 消息，因此所有模式消息的操作码都是 B（所以消息写成“&HBn”，或者“1011nnnn”）。通道模式消息的第一个数据字节是 121~127（即 &H79~7F）。

通道模式消息决定一个乐器怎样处理 MIDI 声部消息。例如，响应所有的消息，只响应对应的通道，完全不响应，或者取决于乐器的本地设定。

在前面所说的状态字节“&HBn”中，n 是通道编号。数据字节的意义见表 6-4。Local Control Off（本地控制关闭）表示断开键盘和合成器的连接（其他的外部设备用来控制声音）。All Notes Off（所有音符关闭）是一个非常方便的命令，特别是在声音中出现了错误的时候，我们就需要它了。Omni 的意思是设备响应所有通道的消息。通常的模式是 OMNI OFF，即各个通道只关注和自身对应的消息，对那些和本通道无关的消息不做响应。Poly 意味着设备同时可以播放多个音符。常用的模式是 POLY ON。

表 6-4 MIDI 模式消息

第一数据字节	描 述	第二数据字节的含义
&H79	重置所有控制器	空；设为 0
&H7A	本地控制	0=off; 127=on
&H7B	所有音符关闭	空；设为 0
&H7C	全向模式关闭	空；设为 0
&H7D	全向模式打开	空；设为 0
&H7E	单声道模式打开（合聚模式关闭）	控制器号
&H7F	合聚模式打开（单声道模式关闭）	空；设为 0

在 POLY OFF（即单音模式）下，可以将表示单音通道编号的参数设置为 0，此时能播放的声部的数目取决于接收器的默认设置，它也可以设置为特定的通道编号。OMNI ON/OFF 以及 Mono/Poly 会产生四种组合。常见的组合是 OMNI OFF, POLY ON。

145

2. 系统消息

系统消息中不带通道编号，表明这个消息不是针对某一个具体的通道，例如同步的时钟信号，录制 MIDI 序列前的位置信息，或者目标设备的一些设置信息。所有系统消息的操作码都以“&HF”开始。根据不同的功能，系统消息可以分成 3 类。

(1) 系统通用消息

表 6-5 列出了这些消息。它们都和时间以及位置相关。Song Position 是通过节拍来设置的。这个消息决定在接收到“播放”的实时消息后，播放一些什么内容。

(2) 系统实时消息

表 6-6 列出了系统实时消息，它们主要是用来处理同步的。

表 6-5 MIDI 系统通用消息

系统通用消息	状态字节	数据字节的数量
MIDI Timing Code	&HF1	1
Song Position Pointer	&HF2	2
Song Select	&HF3	1
Tune Request	&HF6	无
EOX(terminator)	&HF7	无

表 6-6 MIDI 系统实时消息

系统实时消息	状态字节
Timing Clock	&HF8
Start Sequence	&HFA
Continue Sequence	&HFB
Stop Sequence	&HFC
Active Sensing	&HFE
System Reset	&HFF

(3) 系统专有消息

最后一种消息，系统专有消息，使得制造商可以扩展 MIDI 标准。在初始代码后，可以插入应用于它们产品的特定消息的流。如表 6-5 所示，系统专有消息通过结束字节“&HF7”来结束。

不过结束字符只是一个可选项，在很多情况下，发送一个新的消息的状态字节就可以结束前一个数据流。

146

#### 6.2.4 通用 MIDI

为了使 MIDI 音乐在不同的机器上具有相同的播放效果，我们应该给同一种乐器使用相同的 patch 编号。例如，patch1 应该用于钢琴，而不应该是短号。为此，通用 MIDI[5]是给乐器分配 patch 编码的一个方案。标准的打击乐器映射也规定了 47 种标准的打击乐音色。乐谱上出现一个特定的“音符”，表明应该发出某一个特定的打击乐的声音。在本书的网站上有标准的通用 MIDI 乐器路径映射表，以及打击乐映射表。

通用 MIDI 还有一些性能要求，比如 MIDI 设备必须支持所有的 16 个通道，必须支持多音色（即每一个通道能同时播放不同的乐器效果），支持复音（即每个通道能够播放不同的声部），并且必须同时支持 24 个动态分配的声部。

##### 通用 MIDI Level2

它是最近发布的对于通用 MIDI 的扩展，同时发布的还有标准 MIDI 文件(Standard MIDI File, SMF) 格式。一个非常有用的扩展是在 MIDI 文件信息中加入文字信息，比如卡拉 OK 歌词，这些歌词能够在音序器上显示。

#### 6.2.5 MIDI 到 WAV 的转换

一些应用程序，比如早期版本的 Premiere，不支持 MIDI 文件格式，它们要求文件必须是 WAV 格式。很多共享软件能够在这两种文件格式间进行转换。一般这些程序都含有一个查找表文件，将 MIDI 消息转换成预定义的 WAV 输出，不过这种转换并不总是能够成功。

### 6.3 音频的量化和传输

为了传输音频，必须将采样的音频信息数字化，现在我们来详细讨论一下数字化的过程。信息被数字化后，能够很方便地存储和传输。我们通过一些完整的例子来说明我们所讨论的内容。

#### 6.3.1 音频的编码

数据量化以及数据转换统称为数据编码(coding)。对音频信号来说，除了使用扩展压缩音频信号的 $\mu$ 律外，还使用一个简单的算法来消除信号中的即时冗余。将当前时刻的信号和前一时刻的信号逐差能够很有效地降低传输的数据量，更重要是，因为现在我们处理的是差值，能使这些值的分布集中在一个更小的范围内。降低了数据的偏差后，可以使无损压缩方法变得更加高效，从而生成一些压缩率很高的位流。我们将在第 7 章介绍这些压缩方法。

一般来讲，音频信号处理中使用脉冲编码调制(Pulse Code Modulation, PCM)来生成量化采样输出。另外一种方法是差分脉冲编码调制(还有一个比较简单但是很高效的变体，叫做增量调制)。更进一步，人们发明了自适应差分脉冲编码调制(Adaptive Differential Pulse Code Modulation, ADPCM)，此外人们还发明了考虑到语音特性的一些版本。在第 13 章将要介绍更复杂的语音模型。

147

#### 6.3.2 脉冲编码调制

##### 1. PCM 概述

声音是模拟信号，声波通过空气传播到我们的耳膜，我们才能够听到声音。我们知道，从模拟信号生成数字信号最基本的技术就是采样和量化。一般我们都使用均匀采样，即我们选择一个采样频率，然后在每一个采样周期中生成一个采样值。

在幅值方向,我们采用量化的方法来完成数字化,即在幅值方向上选取一些离散点,然后将每个间隔中得到的值重新映射到一个表现输出层上。间隔的边界集合有时称为判定边界(decision boundary)。那些表现值称为重构层(reconstruction level)。

映射到同一个输出级别的量化器输入间隔的边界形成编码器映射(coder mapping),量化器输出值的表现值为解码器映射(decoder mapping)。因为是在做量化,所以需要对采样值确定一个量化精度。最后我们可能要做压缩,给那些出现频率最高的信号值赋一个较小的值来实现压缩。

每种压缩方案都要经历3个阶段:

1) **变换(Transformation)** 将输入数据转换为一个更易于压缩或压缩效果更好的表示。例如在预测编码中(将在后面的章节中讨论),可以通过已有的信号值,来预测下一个信号值,然后传送预测误差。

2) **失真(Loss)** 我们需要引入信息失真的概念。失真主要在量化这一步产生。这里我们只能使用有限数量的重构层,其数量比原始信号值的数量小很多。因此量化过程中必然会损失一部分信息。

3) **编码(Coding)** 我们需要给每一个输出层设定一个码字(通过它来生成二进制流)。码字可以是定长的也可能是变长的,例如霍夫曼编码就是变长码字(将在第7章中讨论)。

对于音频信号,我们首先研究的是PCM。通过它人们想到了无损预测编码以及DPCM方案,它们用的都是差分编码。最后我们研究自适应版本ADPCM,我们期望通过它达到更高的压缩率。

PCM是我们在采样和编码时使用的一种标准方法。称为脉冲是因为从工程师的眼光看来,生成的数字信号可以看成一串垂直脉冲。CD文件中音频采样就是PCM的例子,采样率为44.1kHz,每个采样点是16位。立体声系统有两个采样通道,总的速率大概是1400kbps。

## 2. 语音压缩中的PCM

在6.1.6节中,我们讨论了压缩扩展,它是一种应用在电话系统中的,对语音信号进行压缩和扩展的技术。在这项技术中,首先使用对语音作 $\mu$ 律转换(在欧洲使用A律转换),主要是对信号作一个对数变换。然后是PCM,使用均匀量化。结果就是我们更加关注语音中低音变化,而忽略了高音部分的细微变化。

假设语音的频率范围是50Hz~10kHz,根据奈奎斯特定理,采样率至少是20kHz。如果使用没有压缩扩展的均匀采样,每个采样点至少需要12位。因此,对于一路音频传输,数据率应该是240kbps。如果使用压缩扩展,可以在得到同样的语音效果下将每个采样点减少到8位,数据率也相应减少到160kbps。但是,在标准的电话系统中,认为我们能够发出的最高频率是4kHz。因此采样率是8kHz,压缩扩展的数据率仅仅是64kbps。

虽然语音压缩相对比较简单,还是要交代两个细节,以便大家能对它有正确的认识。第一,因为我们最多只考虑了4kHz的声音,其他的声音都被认为是噪声。因此在输入模拟信号的时候,需要将其中的高频成分清除掉。我们使用一个带限滤波器来完成这个工作,它能够阻断高频信号以及极低频的信号。没有被清除的频带正是我们想要保留的。这种滤波器也称为带通滤波器。

第二,当我们已经成功地得到了脉冲信号,如图6-13a所示,我们仍然需要执行数/模转换,然后重新生成模拟信号。但是我们得到的信号是阶梯形的信号(如图6-13b所示)。这种离散的信号不仅仅包含原始信号的频率成分,而且因为信号中有不平滑的拐角,所以根据傅里叶分析的理论,这个信号中有无穷多的高频信号。我们知道这种信号是后来引入的,所以我们在数/模转换器的输出端安装一个低通滤波器,它只允许低于原始信号频率最大值的信号通过。图6-14显示了编码和解码电话信号的方案。因为低通滤波器的作用,输出信号变得平滑,如图6-13c所示。为了简单起见,图6-13中没有显示压缩扩展的效果。

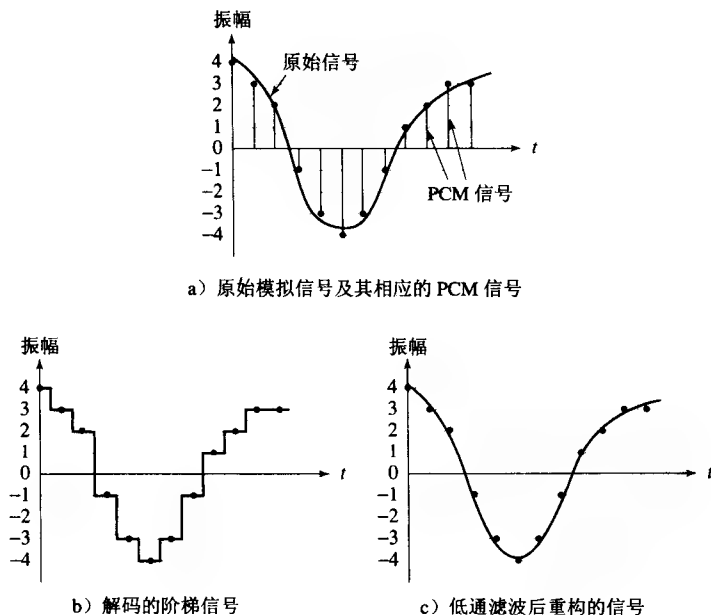


图 6-13 PCM

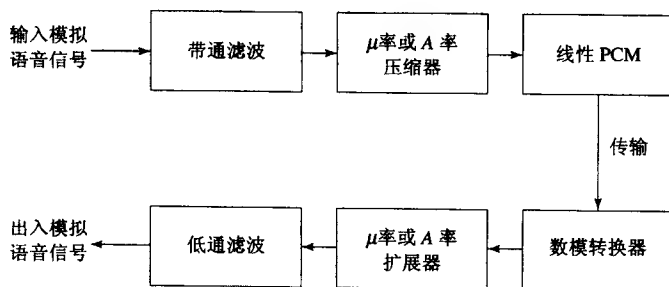


图 6-14 PCM 信号的编码和解码

A 律或 $\mu$ 律 PCM 编码被国际电报电话咨询委员会 (CCITT) 的标准 G.711 采用, 用于数字电话。CCITT 的这一标准现在包含到 ITU 的新标准中。

### 6.3.3 音频的差分编码

音频不仅仅可以通过 PCM 编码来存储, 还可以用差分编码的形式存储。最起码, 差分值都比原来的信号值小, 可以使用更小的空间来存储。

求取差分的一个优点是差分信号的分布图比原始信号的分布图更加集中。例如, 考虑一种极端情况, 按照固定斜率递增的信号分布图是均匀的。然而该信号导数的分布图 (即相连采样点之间的差值) 在原信号的斜率值那一点会有一个峰值。

一般来讲, 如果一个和时间相关的信号随时间变化存在一些临时冗余, 那么把当前信号值和前一个信号值相减产生的差分值将会在分布图上产生一些峰值, 在 0 附近会有最大值。因此, 如果我们给差分值分配码字, 可以给出现频率高的值分配较短的码字, 给出现频率低的值分配较长的码字。

下面我们将首先介绍无损编码。当我们进行量化的时候，必然会带来信息失真。如果我们不使用量化，也是可以进行压缩的，通过逐差的方法，减小信号值的变化范围，这样跟原始信号相比，就会有压缩效果。第7章将介绍一些比较复杂的无损压缩方法，我们先看一种比较简单的方法。使用量化，预测编码变成了DPCM，这是一种有损编码。我们也会介绍那种方法。

### 6.3.4 无损预测编码

预测编码最基本的想法就是传输差分值。我们预测下一个采样值和当前采样值相等，我们不发送具体的采样值，而是发送采样值和预测值的误差。如果我们预测下一个采样值和前一个采样值相等，那么误差就是前一个采样点和后一个采样点之间的差。我们还可以使用更加复杂的预测策略。

但是，注意一个问题。如果整数采样值的范围是0~255，那么差分值的范围是-255~255。所以我们不得不把动态范围（dynamic range）扩大一倍。我们传送差之前需要更多的位。不过我们可以使用一个小技巧来解决这个问题。

简单一点讲，预测编码包括求差值以及传输差值两部分。首先使用6.3.2节介绍的一个PCM系统。我们应该注意到一个整数的差值依然是整数。下面我们用形式化的方法来描述我们的工作。用 $f_n$ 表示信号值， $\hat{f}_n$ 表示预测值， $e_n$ 表示实际信号和预测信号之间的差。

$$\begin{aligned}\hat{f}_n &= f_{n-1} \\ e_n &= f_n - \hat{f}_n\end{aligned}\tag{6.12}$$

我们当然希望差值 $e_n$ 能尽量小，因此我们也就希望预测值能尽量接近真实值。对于某个信号序列，用以前的一些值（如 $f_{n-1}, f_{n-2}, f_{n-3}$ ）组成的函数能够更好地预测 $f_n$ 。一般来说，人们用一个线性函数来进行预测。

$$\hat{f}_n = \sum_{k=1}^{2 \sim 4} a_{n-k} f_{n-k}\tag{6.13}$$

一般要对预测函数得到的预测值进行取整，求得一个整数预测值。在公式中，我们需要设置系数 $a_{n-k}$ ，所以我们可以自行的调整这些参数（参见6.3.7节）。

逐差的主要目的是为了让采样值的分布更加集中。例如，图6-15a显示了1秒钟的采样语音，采样频率是8kHz，每个采样点的精度是8位。

151

如图6-15b所示，这样采样点集中分布在0周围。图6-15c显示了对应的语音信号差分值的分布图。和原始采样值相比，差分值在0周围的聚集度更高。因此，按照较短码字分配给出现频率高的值的原則，我们将最短的码字赋给0以节约存储空间。对差值使用这种编码方法比对原始值使用这种编码手段效果要好很多。如果使用更好的预测手段，其效果也会比仅仅使用前一个采样值作预测值更好。

现在，我们还有一个问题没有解决，如果一个差分值集合包含一些特别大的差分值，我们应该怎么处理？为了解决这个问题，我们需要在我们的差分值列表中定义两个新的码字Shift-Up和Shift-Down，我们用SU以及SD来指代它们。它们在编码中会取两个特殊的值。

假设采样值的范围是0~255，差分值的范围是-255~255。我们定义SU、SD代表增加/减少32。那么，我们就可以为一组有限的信号差值-15~16定义码字。差分值原始取值范围是-255~255，在-15~16范围内的信号差分值可以按照前面讨论的方法编码，如果一个信号差分值不在这个范围内，那么我们用一系列的递增/递减再加上一个-15~16范围内的值来进行转换。例如100可以写成SU、SU、SU、4。



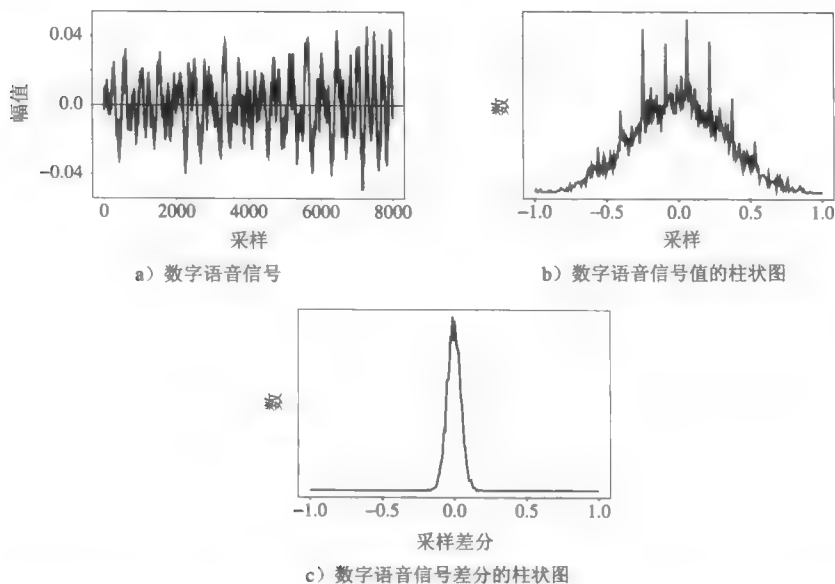


图 6-15 差分集中柱状图

无损预测编码的最主要的特点是无损。也就是说,解码后获得信号应该和原始信号值一致。用一定的篇幅来讨论这个内容对我们理解本书会更有帮助,所以我们在这里进行详细讨论(我们不使用非常复杂的方案,但是我们会给大家显示完整的计算过程)。举一个简单的例子,假设我们通过下面的公式来求预测值  $\hat{f}_n$ :

$$\hat{f}_n = \left\lfloor \frac{1}{2}(f_{n-1} + f_{n-2}) \right\rfloor \quad (6.14)$$

$$e_n = f_n - \hat{f}_n$$

我们实际传输的是误差值  $e_n$ 。

我们来考虑一个特例。假设我们要给序列  $f_1, f_2, f_3, f_4, f_5 = 21, 22, 27, 25, 22$  进行编码。为了进行预测编码,我们首先要设定一个信号值  $f_0$ , 我们把它设成和  $f_1$  相等。首先要传输的值是无需进行编码的初始值。事实上,每一种编码机制都需要头信息的开销。

然后,开始传输第一个误差值  $e_1 = 0$ 。接下来是

$$\begin{aligned} \hat{f}_2 &= 21, \quad e_2 = 22 - 21 = 1 \\ \hat{f}_3 &= \left\lfloor \frac{1}{2}(f_2 + f_1) \right\rfloor = \left\lfloor \frac{1}{2}(22 + 21) \right\rfloor = 21 \\ e_3 &= 27 - 21 = 6 \\ \hat{f}_4 &= \left\lfloor \frac{1}{2}(f_3 + f_2) \right\rfloor = \left\lfloor \frac{1}{2}(27 + 22) \right\rfloor = 24 \\ e_4 &= 25 - 24 = 1 \\ \hat{f}_5 &= \left\lfloor \frac{1}{2}(f_4 + f_3) \right\rfloor = \left\lfloor \frac{1}{2}(25 + 27) \right\rfloor = 26 \\ e_5 &= 22 - 26 = -4 \end{aligned} \quad (6.15)$$

152  
153

大家可以看见, 误差值集中在 0 的附近, 所以编码 (给信号值分配码字) 会有较好的效果。图 6-16 是这种编码系统的示意图。注意, 预测器生成预测值  $\hat{f}_n$ 。而这个预测值总是基于  $f_{n-1}, f_{n-2}, \dots$  因此在预测器中需要配有内存。在预测器中, 至少需要一个电路能够将前一个信号值  $f_{n-1}$  延时一个操作周期。

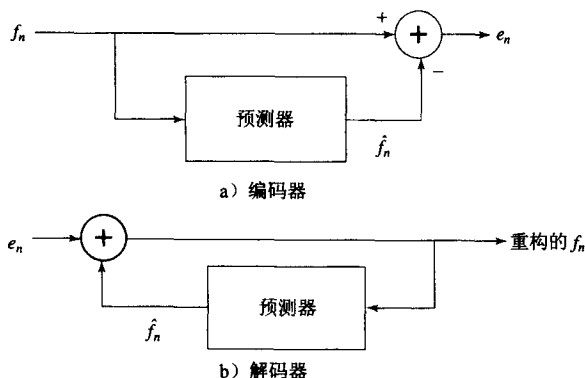


图 6-16 预测编码的示意图

### 6.3.5 DPCM

差分脉冲编码调制 (DPCM) 和预测编码类似, 只是它含有一个量化步骤。量化步骤和 PCM 中的量化步骤类似, 可以是均匀量化, 也可以是非均匀量化。有一种 Lloyd-Max 量化法, 它通过求解误差项的最小方差值, 求得最好的非均匀量化坐标值集合, 这种方法是用 Stuart Lloyd 以及 Joel Max 来命名的。

我们在这里约定一些记号。我们记原始信号为  $f_n$ , 预测信号为  $\hat{f}_n$ , 量化后重构的信号为  $\tilde{f}_n$ 。DPCM 所做的操作流程是, 首先生成预测值, 然后用原始信号值减去预测信号值生成误差值  $e_n$ , 接着将误差值量化, 得到量化后的误差值  $\tilde{e}_n$ 。可以用如下的方程来描述 DPCM:

$$\begin{aligned}
 \hat{f}_n &= \text{function\_of}(\tilde{f}_{n-1}, \tilde{f}_{n-2}, \tilde{f}_{n-3}, \dots) \\
 e_n &= f_n - \hat{f}_n \\
 \tilde{e}_n &= Q[e_n] \\
 &\quad \text{传输 } \text{codeword}(\tilde{e}_n) \\
 &\quad \text{重建: } \tilde{f}_n = \hat{f}_n + \tilde{e}_n
 \end{aligned} \tag{6.16}$$

我们使用熵编码 (例如我们将在第 7 章中讨论的赫夫曼编码) 来生成量化后的误差值  $\tilde{e}_n$  的对应码字。

请注意, 预测都是基于量化重构后的信号值的。这么做的主要原因是, 这样在编码阶段使用的信息在解码阶段同样能够得到。如果我们在预测的时候, 错误地用原始信号值  $f_n$  而不是预测值  $\hat{f}_n$ , 那么量化误差就会被累积起来, 使得最后误差不再在 0 附近分布。

编码/解码的主要作用是生成离散化的、可重构的信号值  $\tilde{f}_n = \hat{f}_n + \tilde{e}_n$ 。我们使用平均方差  $[\sum_{n=1}^N (\tilde{f}_n - f_n)^2] / N$  来衡量信号的“失真率”, 我们也经常看到信号值位数和失真率的关系图。Lloyd-Max 量化产生的失真比均匀量化效果要小。

对于一些信号, 我们需要选择量化的步长, 使得它们能够和信号值的上下界相匹配。即使我

154

们使用的是均匀量化，这样的工作也是有效果的。在语音处理中，我们可以通过求得一段语音的平均值、方差，然后再调整量化步长来得到合适的步长。也就是说，在时刻  $i$ ，我们取一段语音，其中含有  $N$  个采样值  $f_n$ ，我们要使得下面公式表示的量化误差最小：

$$\min \sum_{n=i}^{i+N-1} (f_n - Q[f_n])^2 \quad (6.17)$$

因为信号的差分分布非常集中，我们可以使用 Laplacian 概率分布函数来描述它。Laplacian 的分布[6]也是非常集中在 0 周围，概率分布函数为  $l(x) = (1/\sqrt{2\sigma^2})\exp(-\sqrt{2}|x|/\sigma)$ ，方差值为  $\sigma^2$ 。一般来说，我们假设信号的差分值  $d_n$  符合 Laplacian 分布，然后为量化器选择合适的量化步长（可以是非均匀的）。量化步长的选择标准是要能使得下面的公式有极小值：

$$\min \sum_{n=i}^{i+N-1} (d_n - Q[d_n])^2 l(d_n) \quad (6.18)$$

这是平方极小值的问题，可以通过 Lloyd-Max 量化器来迭代求解。

图 6-17 是 DPCM 编码/解码器的示意图。由于图看上去很普通，所以有一些特点很容易被忽视。第一，预测器使用的是重构的、量化的信号值  $\tilde{f}_n$ ，而不是真实的信号值  $f_n$ 。也就是说，在编码器的预测模块中已经实现了解码的功能。量化器可以使用均匀量化也可以使用非均匀量化。

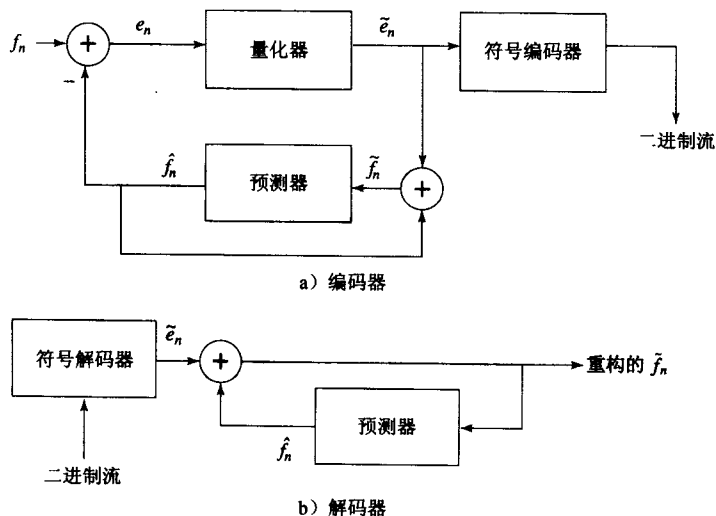


图 6-17 DPCM 的示意图

在示意图中标记“符号编码器”的方框一般指霍夫曼编码器，我们将在第 7 章中详细介绍这个步骤。在生成预测值  $\tilde{f}_n$  时，我们需要使用以前的一些预测值  $\tilde{f}$ ，因此我们就需要缓存这些预测值。值得注意的是，量化噪声  $f_n - \tilde{f}_n$  和对差分值量化产生的误差  $e_n - \tilde{e}_n$  是相等的。

使用真实数据有助于我们更好地理解编码过程。假如我们使用下面的预测器：

$$\hat{f}_n = \text{trunc}[(\tilde{f}_{n-1} + \tilde{f}_{n-2})/2] \quad (6.19)$$

$e_n = f_n - \hat{f}_n$  是一个整数。

使用下面的公式来进行量化：

$$\begin{aligned}\tilde{e}_n &= Q[e_n] = 16 * \text{trunc}[(255 + e_n)/16] - 256 + 8 \\ \hat{f}_n &= \hat{f}_n + \tilde{e}_n\end{aligned}\quad (6.20)$$

首先我们注意到, 误差的范围是-255~255, 也就是说, 误差可能有 511 种取值。量化器把误差值划分成 32 块, 每块在原始信号中跨越 16 个取值范围。

表 6-7 给出了输入信号值的输出值: 4 位编码以阶梯函数的形式映射到 32 个重构值(注意, 最后一个区间只有 15 个值, 而不是 16 个值)。

举一个信号流的例子。考虑如下信号值集合:

$$\begin{array}{ccccc} f_1 & f_2 & f_3 & f_4 & f_5 \\ 130 & 150 & 140 & 200 & 230 \end{array}$$

我们假设在数据流中  $f$  取值和信号值  $f_1$  一致, 也就是  $f=130$ 。量化的初始误差为  $\tilde{e}_1 \equiv 0$ , 这样我们的第一个重构值为  $\hat{f}_1 = 130$ 。后续的值列举在下面的表格中(方框中的是初始值):

$$\begin{array}{rcll} \hat{f} & = & \boxed{130} & 130, \quad 142, \quad 144, \quad 167 \\ e & = & \boxed{0} & 20, \quad -2, \quad 56, \quad 63 \\ \tilde{e} & = & \boxed{0} & 24, \quad -8, \quad 56, \quad 56 \\ \tilde{f} & = & \boxed{130} & 154, \quad 134, \quad 200, \quad 223 \end{array}$$

解码的时候, 我们同样假设初始值  $\tilde{f}$  和  $\hat{f}_1$  一致, 这样第一个重构值就是  $\tilde{f}_1$ 。只要我们使用相同的预测规则, 我们接收到的是误差值  $\tilde{e}_n$ , 这样重构值  $\tilde{f}_n$  和编码阶段是一致的。

### 6.3.6 DM

DM 是 Delta Modulation (增量调制) 的缩写, 它是 DPCM 的简化版本, 常常作为一个快速的模/数转换器。为了保持内容的完整性, 下面将介绍这种方法。

#### 1. 均匀 Delta DM

DM 的主要思想是仅仅使用唯一的量化值, 该值可以是正数也可以是负值。这种只使用一位的编码使得原始信号编码后呈阶梯形状。相关方程组如下所示:

$$\begin{aligned}\hat{f}_n &= \tilde{f}_{n-1} \\ e_n &= f_n - \hat{f}_n = f_n - \tilde{f}_{n-1} \\ \tilde{e}_n &= \begin{cases} +k & \text{如果 } e_n > 0, \text{ 其中 } k \text{ 是常数} \\ -k & e_n \leq 0 \end{cases} \\ \tilde{f}_n &= \hat{f}_n + \tilde{e}_n\end{aligned}\quad (6.21)$$

注意, 这样的预测方法只有一个时延。

我们现在来举一个具体数值的例子。假设我们有如下的一系列信号值:

$$\begin{array}{cccc} f_1 & f_2 & f_3 & f_4 \\ 10 & 11 & 13 & 15 \end{array}$$

表 6-7 DPCM 量化器的重构级别

$e_n$ 的范围	量 化 值
-255~-240	-248
-239~-224	-232
$\vdots$	$\vdots$
-31~-16	-24
-15~0	-8
1~16	8
17~32	24
$\vdots$	$\vdots$
225~240	232
241~255	248

我们也设初始的重构值为  $\hat{f}_1 = f_1 = 10$ 。

假设我们把步长值设为  $k=4$ ，就能得到下面的值：

$$\begin{aligned}\hat{f}_2 &= 10, & e_2 &= 11-10=1, & \tilde{e}_2 &= 4, & \tilde{f}_2 &= 10+4=14 \\ \hat{f}_3 &= 14, & e_3 &= 13-14=-1, & \tilde{e}_3 &= -4, & \tilde{f}_3 &= 14-4=10 \\ \hat{f}_4 &= 10, & e_4 &= 15-10=5, & \tilde{e}_4 &= 4, & \tilde{f}_4 &= 10+4=14\end{aligned}$$

我们发现重构值 10、14、10、14 与原始值 10、11、13、15 的偏离并不是太大。

可以看出，在信号基本上维持常数值而很少变动时，使用 DM 会有很好的效果，但是如果信号变化很剧烈，效果就不理想了。解决这种问题的一个简单的方法就是提高采样频率，可以提到奈奎斯特采样率的若干倍。通过这种方法，DM 可以成为一种比较简单但也比较有效的模/数转换器。

## 2. 自适应 DM

如果信号波形非常陡峭，那么阶梯状的逼近就不会有很好的效果。处理陡峭波形的一个简单方法就是自适应地变化步长值  $k$ ，也就是说，根据信号当前的特点来设定步长值  $k$ 。

### 6.3.7 ADPCM

自适应 DPCM 的思想就是自动调整编码机制使得它能更好地适应输入值。DPCM 编码器最主要有两个部分：量化器和预测器。在上文提及的自适应 DM 中，我们调整量化器的步长去适应输入。在 DPCM 中，我们可以自适应地调整量化器，包括调整步长以及在非均匀量化中调整判别边界。

我们可以通过两种方式来实现这种目标：使用输入信号的特点（我们称为前向自适应量化），或者使用量化输出信号的特点。如果量化误差太大，我们需要修改非均匀 Lloyd-Max 量化器（我们称为后向自适应量化）。

我们也可以使用前向或者后向的方法来调整预测器。一般称自适应变化预测系数的编码方式为自适应预测编码（Adaptive Predictive Coding, APC）。这种方法的具体实现非常有趣。前面讲过，预测器一般是已有的重构量化信号值  $\tilde{f}_n$  的线性函数。我们称一个预测器使用的以前的预测值的数目为预测器的阶（order）。例如，如果我们使用了  $M$  个以前的预测值，我们在下面的预测器中需要  $M$  个系数  $a_i (i=1, \dots, M)$ ：

$$\hat{f}_n = \sum_{i=1}^M a_i \tilde{f}_{n-i} \quad (6.22)$$

从公式中我们可以看出，系数  $a_i$  与以前量化的值相乘。如果我们要修改这组系数，就会遇到一个难题，我们需要求解一组及其复杂的方程组。假设我们使用最小二乘法来求解这个极小值问题，希望能够求解到  $a_i$  的最优解：

$$\min \sum_{n=1}^N (f_n - \hat{f}_n)^2 \quad (6.23)$$

在公式里，我们需要为当前语音段中大量的采样点  $f_n$  求和。因为  $\hat{f}_n$  是量化值，我们还有一些其他难题要解决。同时，我们也需要实时保证量化过程的正确性，以适应信号不断变化的特点，这使得问题更加复杂。

相反，我们转而求解一个简单一点的问题，我们在最小二乘法中，不使用预测重构值  $\tilde{f}_n$ ，而使用原始信号值  $f_n$ 。这使得问题变得容易解决，因为如果带入系数  $a_i$ ，我们要求解的问题是

$$\min \sum_{n=1}^N (f_n - \sum_{i=1}^M a_i f_{n-i})^2 \quad (6.24)$$

注意其中  $a_i$  的取值, 将其中的一部分设置为 0, 我们就得到一个易解的  $M$  个方程的线性系统 (这组方程被称为 Wiener-Hopf 方程组)。

这样, 我们就找到了一种自适应改变预测器的方法。对于语音信号来说, 考虑一组语音值是很常见的, 就像对一幅图像进行编码, 我们可以自适应地修改预测器、量化器, 或者两者同时改动。如果我们使用 8kHz 来采样, 常把 128 个采样点作为一个语音段——大概是 16 毫秒的语音。图 6-18 显示了 ADPCM 的编码和解码[7]的示意图。

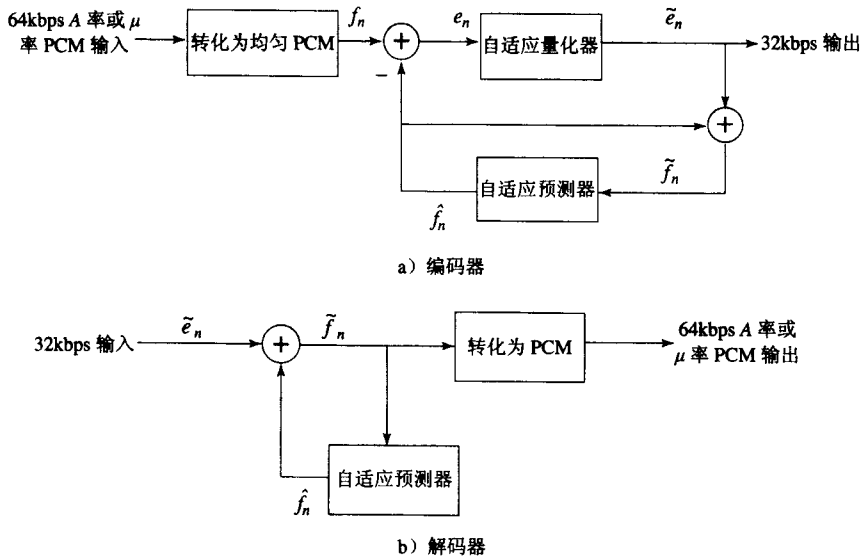


图 6-18 ADPCM 示意图

## 6.4 进一步探索

研究人员在使用音频帮助视力有缺陷的人士方面正在进行大量的工作。一项技术是使用音频来协助显示 HTML 结构, [8, 9, 10]这几篇论文中提供了一些富有创造性的想法。

Pohlmann 的书[2]是了解数字化、SNR、SQNR 的极佳的参考资料。本书网站上第 6 章的 Further Exploration 部分中将详细介绍音频量化中的  $\mu$  律。下面是一些其他一些有用的链接:

- 关于使用 FM 合成语音的讨论。
- 关于语音文件格式的一个比较全面的列表。
- 一个关于不同 CD 音频文件格式的详细描述。各种音频文件之间或多或少有一些差别。主要的音乐格式是 red book audio。
- 一个通用 MIDI 乐器的 Patch Map, 以及一些通用 MIDI 打击乐的基调 Map。
- 一个关于 MIDI 音乐合成和波形表音乐合成方法的很好的教程。
- 一个关于使用 Java 来解码 MIDI 数据流的链接。
- 一个很好的多媒体/声音页面, 提供了很多的 Internet 上的声音和音乐资源。
- 一个面向表演艺术的站点, 其中有很多的关于音频的资料, 内容包括许多术语的定义, 信号处理, 以及声音体验。

## 6.5 练习

1. 我的旧 SoundBlaster 声卡是 8 位卡。
  - (a) 这里的 8 位是指什么是 8 位?
  - (b) 它最好能提供什么样的 SQNR?
2. 如果一个耳塞能把噪声降低 30dB, 那么它一共降低了多少强度(能量)?
3. 因为存储介质(如磁带)以及音频放大器的频率响应函数, 音频输出时在听觉所及范围的上、下边缘的信息损失是不可避免的。
  - (a) 如果对于中音频率, 输出电压是 1V, 那么对 18kHz 频率, 在损失 3dB 以后输出电压是多少?
  - (b) 为了补偿这些损失, 听众可以调整均衡器在不同频率上的增益(从而调整了输出)。如果损耗是 -3dB, 均衡器在 18kHz 的增益是 6dB, 那么输出电压是多少?(提示: 假设  $\log_{10}(2)=0.3$ 。)
4. 假设采样频率是真实频率的 1.5 倍。那么假频是多少?
5. 在一个拥挤的房间中, 尽管噪声很大, 我们依然可以分辨并且听懂周围人的声音, 这称为鸡尾酒派对效应(cocktail-party effect)。具体的原理是我们的听觉器官能够根据左右耳朵的声音相位差判断出声源的位置(binaural auditory perception)。在单声道中, 如果噪声很大, 我们无法听出周围人的声音。说说你认为卡拉 OK 系统是如何工作的? 提示: 在商用音乐录音的混音系统中, 每种乐器的左右声道的 pan 参数不同。因此对一种乐器, 它的左右通道都被标定了, 请问演唱者的声道的时间, 如何记录才能更加容易地去除演唱者的声音(通常的做法)。
6. 一个信号  $V$  的动态范围(dynamic range)是它的最大绝对值和最小绝对值的比率。从某种意义上讲, 一个信号动态范围反应一个信号的质量。它也能够反映出我们需要多少位才能把量化噪声限制在一个可以接受的范围内。例如, 我们可能希望量化噪声的振幅比  $V_{min}$  要小。假设信号动态范围是 60dB, 我们使用 10 位能够满足要求吗? 16 位呢?
7. 动态范围的定义是  $V_{max}/V_{min}$ , 电话系统的动态范围是 256。若使用均匀量化, 那么应该使用多少位来编码, 才能使得量化噪声的振幅小于电话中能够听到的最轻的声音?
8. 感知不一致性(perceptual nonuniformity)是一个常用术语, 它说明了人对声音的感受能力不是线性分布的。也就是说, 当音频信号的某个参数变化时, 人不是按照比例感受到对应的变化。
  - (a) 简要介绍至少两种人在音频感受上的感应能力非均匀分布。
  - (b) 哪一种可以使用 A 律(或者  $\mu$  律)来逼近? 它为什么能够提高量化的效果?
9. 绘制一幅示意图, 显示一个频率为 5.5kHz 的正弦函数使用 8kHz 频率来采样(在图表中要绘制 9 个采样点)。绘制一条 2.5kHz 的假频, 在图中应该可以看出在 8 个采样间隔中, 真实信号的 5.5 个周期刚好和干扰频率的 2.5 个周期一致。
10. 假设一个信号含有 1、10 以及 21kHz 的成分, 使用 12kHz 的速率来采样(然后使用最高频率为 6kHz 的抗干扰滤波器来处理), 输出信号中会含有哪些频率成分?(提示: 大部分输出都会含有干扰现象)。
11. (a) 一个 MIDI 消息能否同时产生多个声部?
  - (b) 利用 MIDI, 能够在一个乐器上一次模拟出多个声部吗? 如果可以, 是怎样实现的?
  - (c) Program Change MIDI 消息是通道消息吗? 这个消息是如何实现的? 根据 Program Change 消息, 通用 MIDI 中一共有多少个乐器? 为什么?
  - (d) 一般来讲, MIDI 消息主要分为哪两种类型? 从具体数据上来讲, 两类消息最主要的区

别是什么？继续给这两大类消息分类，还可以分为什么类型？

12. (a) 使用文字而不是 16 进制值给出一个 MIDI 声音消息的例子。
- (b) 描述这则消息的 Assembler 声明部分。
- (c) Program Change 消息的主要功能是什么？假设 Program Change 写成 16 进制是 “&HC1”，那么指令 “&HC103” 的主要功能是什么？
13. 在 PCM 中，若使用 8kHz 采样，那么延时是多少？一般来讲，在信号采样、处理、分析的算法中，延时是指算法所需要消耗的时间。
14. (a) 假设我们使用下面的预测器：

$$\hat{f}_n = \text{trunc} \left[ \frac{1}{2} (\tilde{f}_{n-1} + \tilde{f}_{n-2}) \right] \quad (6.25)$$

$$e_n = f_n - \hat{f}_n$$

并且我们使用量化方程 (6.20)，如果输入信号值如下：

20, 38, 56, 74, 92, 110, 128, 146, 164, 182, 200, 218, 236, 254

DPCM 编码器输出如下（不含熵编码）：

20, 44, 56, 74, 89, 105, 121, 153, 161, 181, 195, 212, 243, 251

图 6-19a 显示了量化重构信号和原信号的轨迹。作为一个编程项目，编写一段代码来验证你的结果。

- (b) 假设在无损编码中，由于我们使用预测器（式 (6.14)）时发生了错误，使用原始信号值  $f_n$  而不是量化信号值  $\tilde{f}_n$ ，所以在解码端，我们获得如下的重构信号：

20, 44, 56, 74, 89, 105, 121, 137, 153, 169, 185, 201, 217, 233

可以看出误差在逐渐加剧。

图 6-19b 显示重构信号的效果越来越差，修改上面的代码来验证上述结论。

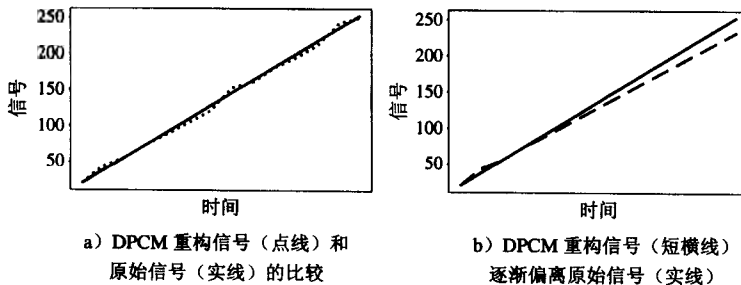


图 6-19 第 14 题的示意图

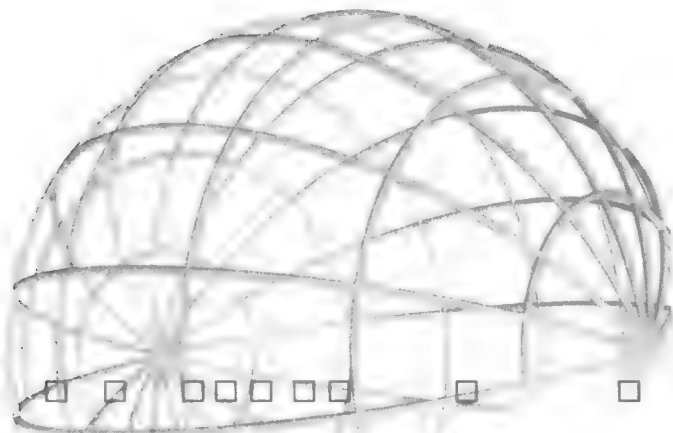
## 6.6 参考文献

- [1] B. Truax, *Handbook for Acoustic Ecology*, 2nd ed., Burnaby, BC, Canada: Cambridge Street Publishing, 1999.
- [2] K.C. Pohlmann, *Principles of Digital Audio*, 4th ed., New York: McGraw-Hill, 2000.
- [3] J.H. McClellan, R.W. Schafer, and M.A. Yoder, *DSP First: A Multimedia Approach*, Upper Saddle River, NJ: Prentice-Hall PTR, 1998.
- [4] J. Heckroth, *Tutorial on MIDI and Music Synthesis*, La Habra, CA: The MIDI Manufacturers



- Association, 1995, [www.harmony-central.com/MIDI/Doc/tutorial.html](http://www.harmony-central.com/MIDI/Doc/tutorial.html).
- [5] P.K. Andleigh and K. Thakrar, *Multimedia Systems Design*, Upper Saddle River, NJ: Prentice-Hall PTR, 1984.
  - [6] K. Sayood, *Introduction to Data Compression*, 2nd ed., San Francisco: Morgan Kaufmann, 2000.
  - [7] Roger L. Freeman, *Reference Manual for Telecommunications Engineering*, 2nd ed., New York: Wiley, 1997.
  - [8] M.M. Blattner, D.A. Sumikawa, and R. Greenberg, "Earcons and Icons: Their Structure and Common Design Principles," *Human-Computer Interaction*, 4: 11-44, 1989.
  - [9] M.M. Blattner, "Multimedia Interfaces: Designing for Diversity," *Multimedia Tools and Applications*, 3: 87-122, 1996.
  - [10] W.W. Gaver and R. Smith, "Auditory Icons in Large-Scale Collaborative Environments," in *Readings in Human-Computer Interaction: Toward the Year 2000*, ed. R. Baecker, J. Grudin, W. Buxton, and S. Greenberg, San Francisco: Morgan-Kaufman, 1990, pp. 564-569.





## 第二部分 多媒体数据压缩

第 7 章 无损压缩算法

第 8 章 有损压缩算法

第 9 章 图像压缩标准

第 10 章 基本视频压缩技术

第 11 章 MPEG 视频编码 I：MPEG-1 和 MPEG-2

第 12 章 MPEG 视频编码 II：MPEG-4、MPEG-7 及更高版本

第 13 章 音频压缩技术基础

第 14 章 MPEG 音频压缩

在这一部分中，我们将讨论数据压缩。数据压缩应该是实现当代多媒体系统的一项最为重要的技术。

第 7 章将介绍无损数据压缩，无损是指数据压缩和重组后与原信号相比没有损失。在保存档案等情况下，需要处理的数据量很大，所以对信息进行压缩非常必要。无损压缩是一种处理方式。

例如，假设我们想做一下 MRI（核磁共振）来检查身体的健康状况。我们当然希望无损地保存这一价格不菲的医疗信息。尽管这种图片的数据量巨大，我们也不愿丢失任何信息。我们可以应用 WinZip 这一常用的无损压缩工具。

另一个例子是对以往的画作进行存档。大师的作品不但要精心拍摄，而且保存时最好不要去掉这么难得的信息，因此也需要无损压缩。

而另一方面，如果是家庭中所看的电影的数据，则可丢失一些信息。如果 PC 不能处理那么多数据，我们要在随意丢失一些数据和利用有损压缩方法有目的地丢失一些信息中选择的话，一定会选择后者。现在，几乎所有我们见到的视频都是以有损方式压缩的。而网上大量的标准 JPEG 格式的图像也都是有损压缩的。

因此，从第 8 章开始，我们讨论有损压缩方式，主要关注离散余弦变换和离散小波变换。这些方法的主要应用是 JPEG 静态图像压缩标准，以及 JPEG 2000，这部分内容在第 9 章介绍。

接着我们继续讨论能够用于动态图像（视频）的数据压缩方法。我们在第 10 章先讨论基本的视频压缩技术。第 11 章以 MPEG-1 和 MPEG-2 为主讨论 MPEG 标准，第 12 章讨论 MPEG-4、MPEG-7 等。音频压缩有其自身的特点，我们在第 13 章讨论基本的音频压缩技术，在第 14 章讨论 MPEG 音频，包括 MP3。

## 第7章 无损压缩算法

### 7.1 简介

多媒体技术的出现使数字图书馆 (digital library) 成为现实。今天, 图书馆、博物馆、电影工作室乃至政府部门都在尝试着将越来越多的数据和档案文件转换成数字的形式。其中, 有些数据 (比如珍贵的书籍和画作) 在保存时不能有任何损失。

假设我们要对国会图书馆的 1 亿 2000 万个左右的馆藏的编目号码进行编码 (如果只考虑书籍的话, 只有 2000 万个)。为什么我们没有简单地把每一个编目号码用一个 27 位的数字来表示, 从而给每个号码赋予一个唯一的二进制编码呢 ( $2^{27} > 120\,000\,000$ ) ?

这样做的主要问题是需要的位过多。事实上, 现在有许多编码技术能够有效地减少表达上述信息所需的总位数。而这里所涉及的过程通常称作压缩 (compression) [1, 2]。

在第 6 章, 我们对音频的压缩方案已经有了初步的认识。在第 6 章, 我们必须首先考虑将模拟信号转换为数字信号的复杂性, 而在这里, 我们的讨论将直接从数字信号开始。举个例子, 尽管我们知道图像采集时使用的是模拟信号, 但是由数字摄像机产生的文件的确是数字化的。长期以来, 人们一直在研究一个更普遍的问题, 即如何对一组任意的符号, 而不仅仅是字节值进行编码。

回到前面的国会图书馆的问题, 我们知道编目号码的某些部分比其他部分出现的频率更高, 因此在编码时给这些出现频率高的部分赋予较少的位数会更加经济。这也就是所谓的变长编码 (VLC), 即在编码时给出现频率较高的符号赋予较少的位数, 反之亦然。这样, 我们就可以使用更少的位数来表示全部的馆藏。

在本章, 我们将学习一些有关信息论的基础知识和几种应用广泛的无损压缩技术。图 7-1 描述了一个通用的数据压缩方案, 其中压缩由编码器实现, 而解压缩由解码器实现。

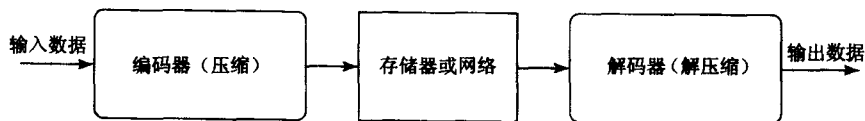


图 7-1 一个通用的数据压缩方案

我们把编码器的输出称作编码 (code) 或码字 (codeword)。中间的媒介可以是数据存储器或是通信/计算机网络。如果压缩和解压缩过程中没有任何信息丢失, 则这种压缩方案就是无损的 (lossless), 否则就是有损的 (lossy)。接下来的几章都是讲述有损压缩算法的, 因为它们通常应用于图像、视频和音频压缩。在本章, 我们将重点介绍无损压缩。

如果压缩前某个数据的总位数为  $B_0$ , 而压缩后该数据的总位数为  $B_1$ , 我们可以定义压缩率 (compression ratio) 为:

$$\text{压缩率} = \frac{B_0}{B_1} \quad (7.1)$$

一般来说, 我们期望任何一个多媒体数字信号编解码器 (编码/解码方案) 都具有远大于 1.0 的压缩率。只要这种方案在计算上是可行的, 那么压缩率越高, 无损压缩方案越好。

## 7.2 信息论基础

根据贝尔实验室的著名科学家 Claude E. Shannon 的研究 [3,4], 一个具有符号集  $S = \{s_1, s_2, \dots, s_n\}$  的信息源 (source) 的熵 (entropy)  $\eta$  定义为

$$\eta = H(S) = \sum_{i=1}^n p_i \log_2 \frac{1}{p_i} \quad (7.2)$$

$$= - \sum_{i=1}^n p_i \log_2 p_i \quad (7.3)$$

其中  $p_i$  是  $S$  中符号  $s_i$  出现的概率。

$\log_2 \frac{1}{p_i}$  表明了包含在  $s_i$  中的信息量 (即由香农[3]定义的所谓的自信息量), 它与对  $s_i$  进行编

码所需的位数<sup>①</sup>相等。举例来说, 如果在一份稿件中出现字符  $n$  的概率为  $1/32$ , 则这个字符包含的信息量是 5 位。换句话说, 需要使用 15 位对字符串  $nnn$  编码。这就是文本压缩中可能的减少数据量的基础, 因为它将使字符编码方案与 ASCII 码表示有所不同, ASCII 码中每个字符使用 8 位来表示。

什么是熵? 科学地说, 熵是对一个系统的无序性 (disorder) 的度量。熵越大, 系统越无序。一般来说, 当我们希望系统更有序时, 我们会给系统增加负熵。比如, 假设我们想对一叠卡片进行排序 (考虑对这些卡片进行冒泡排序, 尽管这不一定是你通常对卡片进行排序的方式)。对于每一次交换或不交换的决策, 我们都将为卡片系统增加 1 位的信息量并且为这叠卡片传送 1 位的负熵。

熵定义包含了这样的思想: 两次决策意味着在以 2 为底的对数中传输两次负熵。一个 2 位的向量可以有  $2^2$  种状态, 而对数将把这个值转换成 2 位的负熵。两次决策会引起两倍的熵改变。

现在, 假设我们希望通过网络来传送这些交换决策。那么我们必须发送 2 位来表示两次决策。如果我们有一个两次决策系统, 那么, 所有这样的传送所需的平均位数也是 2 位。如果我们愿意, 我们可以把 2 位系统中可能的状态数看做 4 个输出结果。每个结果具有  $1/4$  的概率。所以平均来说, 传送每个结果所需发送的位数为  $4 \times (1/4) \times \log_2(1/(1/4)) = 2$  位, 这个结果不足为奇。要传送两次决策的结果, 我们需要传送 2 位。

但是如果某个结果的概率高于其他的结果, 则我们发送的平均位数将会有所不同。(在卡片已经部分有序时可能发生这种情况, 这时不交换的概率要高于交换的概率。) 假设我们的四个状态中某个状态的概率为  $1/2$ , 而其他三个状态发生的概率分别为  $1/6$ 。为了对平均发送位数模型进行扩展, 我们不得不借助于 2 的非整数次幂来表示概率。然后我们可以使用对数来寻求到底需要多少位来传送信息内容。根据式 (7.3), 我们需要传送  $(1/2) \times \log_2(2) + 3 \times (1/6) \times \log_2(6) = 1.7925$  位, 比 2 位要少。这反映出如果能够对我们的四个状态进行编码, 并且使出现概率最高的一个状态用较少的位数来传送, 那么我们可以用更少的位来传送平均值。

熵定义旨在从数据流中识别出经常出现的符号作为压缩位流中的候选短 (short) 码字。正如前面所描述的, 我们使用变长编码方案进行熵编码, 即给频繁出现的符号分配能够快速传输的码字, 而给不经常出现的符号分配较长的码字。比如, 在英语中  $E$  频繁出现, 所以我们应该给  $E$  赋予比其他字母 (如  $Q$ ) 更短的码字。

① 因为我们在上面的定义中选择 2 作为对数的底, 所以信息的单位是位, 这对于数字计算机中的二进制编码表示自然是最合理的。如果对数的底是 10, 则单位是哈特利 (hartley); 如果底是  $e$ , 则单位是奈特 (nat)。

在接收数据流中出现频率低的符号时,这种令人惊奇的一面在(7.3)的定义中有所反映。如果一个符号很少出现,则它出现的概率 $p_i$ 就较低(例如,1/100),它的对数是一个较大的负数。这反映出它在编码时应用较长的位串。式(7.3)中对数外面的概率 $p_i$ 表明在一个较长的流中,符号出现的平均频率与它们出现的概率相等。这种权重应该将见到的某一特定符号时由“惊奇”成分给出的或长或短的信息内容相乘起来。

下面看另外一个例子。如果信息源 $S$ 是一个灰度数字图像,则 $s_i$ 是取值从 $0 \sim (2^k - 1)$ 的一个灰度亮度值,其中 $k$ 是用于表示未压缩图像中每个像素的位数。因为一般使用8位来表示一个像素,这样每个像素恰好可以用1字节表示,所以这个范围通常是 $[0, 255]$ 。图像直方图(在第3章中讨论过)是一种用来计算一幅图像中灰度亮度为 $i$ 的像素的概率 $p_i$ 的有效方法。

从式(7.3)还可以看出,如果一个符号出现的频率为0,则我们就不需将其计入熵中,因为0没有对数。

169

图7-2a给出了灰度亮度呈均匀分布的一幅图像的直方图。也就是说,对于任意的 $p_i$ , $p_i = 1/256$ 。因此,这幅图像的熵为

$$\eta = \sum_{i=0}^{255} \frac{1}{256} \cdot \log_2 256 = 8 \quad (7.4)$$

在式(7.3)中可以看到,熵 $\eta$ 是项 $\log_2 \frac{1}{p_i}$ 的加权和;因此,它代表了源 $S$ 中每个符号所包含的平均信息量。对于一个无记忆的源 $^\ominus S$ ,熵 $\eta$ 代表了表示 $S$ 中每个符号所需的最小平均位数。换句话说,它指明了对 $S$ 中每个符号进行编码所需的平均位数的下界。

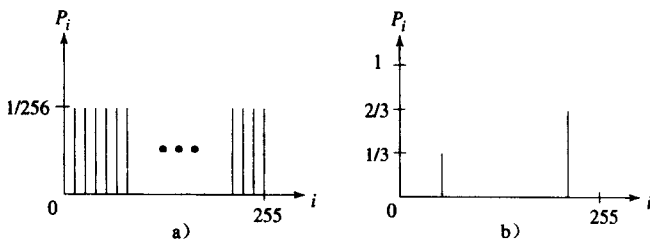


图 7-2 两个灰度图像的直方图

如果我们使用 $\bar{l}$ 来表示编码器生成的码字的平均长度(单位为位),那么香农定理说明了熵是我们所能够取得的最优值(在某种条件下):

$$\eta \leq \bar{l} \quad (7.5)$$

编码方案旨在尽可能地接近这个理论的下界。

我们可以在上面均匀分布的例子中观察到一个有趣的结果: $\eta = 8$ ,即表示每个灰度亮度所需的最小平均位数至少是8。也就是说,这幅图像不可能再被压缩。在生成图像的上下文中,这对应着“最坏的情况”,此时相邻两个像素没有任何相似性。

图7-2b给出了另一幅图像的直方图,其中1/3的像素是比较暗的,而2/3的像素比较亮。这幅图像的熵为:

$^\ominus$  信息源具有独立的分布,意味着当前符号的值并不依赖于前面出现的符号的值。

$$\eta = \frac{1}{3} \cdot \log_2 3 + \frac{2}{3} \cdot \log_2 \frac{3}{2} = 0.33 \times 1.59 + 0.67 \times 0.59 = 0.52 + 0.40 = 0.92$$

170 一般而言, 如果概率分布比较平稳则熵较大, 如果分布有尖峰, 则熵较小。

### 7.3 游长编码

对于有记忆的信息源, 游长编码 (Run-Length Coding, RLC) 利用了信息源中的记忆。这是数据压缩中最简单的一种形式。其基本思想是, 如果我们要压缩的信息源中的符号具有这样的性质, 即同一个符号常常形成连续的片段出现, 那么我们可以对这个符号以及这个片段的长度进行编码, 而不是对片段中的每个符号单独编码。

举例来说, 考虑一个具有单调区间的二值图像 (黑白图像)。利用游长编码算法可以对这个信息源进行高效地编码。事实上, 因为只有两个符号, 所以我们并不需要在每一个行程的开始对任何符号编码, 相反, 我们假设每个行程总是开始于一个特殊的颜色 (黑或白), 然后只需对每个行程的长度编码即可。

上面描述的是一维的游长编码算法。这种算法的二维变体常常用于二值图像编码。这种算法利用图像中前面一行的已编码的行程信息对当前行的行程进行编码。算法的完整描述可参看[5]。

### 7.4 变长编码

由于熵可以表示一个信息源  $S$  的内容, 所以出现了一系列通常被称作熵编码 (entropy coding) 的编码方法。前面介绍过, 变长编码 (Variable-Length Coding, VLC) 就是目前最著名的一种熵编码。在本节, 我们将学习香农-凡诺算法、赫夫曼编码和自适应赫夫曼编码。

#### 7.4.1 香农-凡诺算法

香农-凡诺算法是由贝尔实验室的 Shannon 和 MIT 的 Robert Fano 独立开发的[6]。为了说明这个算法, 我们假设对单词 HELLO 中出现的几个字符进行编码。每个字符出现的频率计数如下:

符号	H	E	L	O
数目	1	1	2	1

香农-凡诺算法的编码步骤可以按照下面的自顶向下的方式描述:

1) 根据每个符号出现的频率对符号进行排序。

2) 递归地将这些符号分成两部分, 每一部分中的符号具有相近的频率, 直到所有的部分都只含有一个符号为止。

171 实现上述过程的一种很自然的方法就是建立一棵二叉树 (binary tree)。按照惯例, 我们给二叉树中的左支赋予 0, 给所有的右支赋予 1。

开始时, 这些符号被排列成 LHEO。如图 7-3 所示, 第一次划分产生两部分: 1) 出现次数为 2 的 L, 记为 L:(2)。2) 出现总次数为 3 的 H, E 和 O, 记为 H,E,O:(3)。第二次划分产生 H:(1) 和 E,O:(2)。最后一次划分产生 E:(1) 和 O:(1)。

表 7-1 总结了上述结果, 给出了单词 HELLO 中的符号、每个符号出现次数、信息内容

$\left( \log_2 \frac{1}{p_i} \right)$ 、结果码字和用来对每个符号进行编码所需的位数。表的最后一行给出了对单词 HELLO 编码所需的总位数。



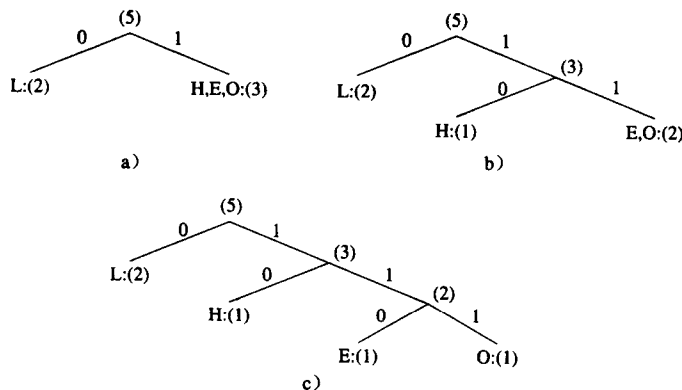


图 7-3 使用香农-凡诺算法对 HELLO 进行编码的编码树

在这个例子中，我们再来回顾一下前面讨论的熵概念：

$$\eta = p_L \cdot \log_2 \frac{1}{p_L} + p_H \cdot \log_2 \frac{1}{p_H} + p_E \cdot \log_2 \frac{1}{p_E} + p_O \cdot \log_2 \frac{1}{p_O}$$
$$= 0.4 \times 1.32 + 0.2 \times 2.32 + 0.2 \times 2.32 + 0.2 \times 2.32 = 1.92$$

表 7-1 在 HELLO 上应用香农-凡诺算法的一个结果

信号	数量	$\log_2 \frac{1}{p_i}$	编码	使用的位数
L	2	1.32	0	2
H	1	2.32	10	2
E	1	2.32	110	3
O	1	2.32	111	3
总位数：10				

这表明对单词 HELLO 中每个字符进行编码所需的最小平均位数应当至少是 1.92。在这个例子中，香农-凡诺算法对每个符号进行编码的平均值为  $10/5 = 2$  位，这与下界 1.92 已相当接近。显然，这个结果令人满意。

应当指出，香农-凡诺算法的结果并不唯一。例如，对上例进行第一次划分时，所有的符号也可以被划分为 L,H:(3)和 E,O:(2)两部分。这时编码结果就变成图 7-4 所示的样子。表 7-2 中的码字也有所变化。另外，这两组码字在有错误出现时可能会呈现出不同的行为。但是巧合的是，对单词 HELLO 编码所需的总位数仍然保持为 10。

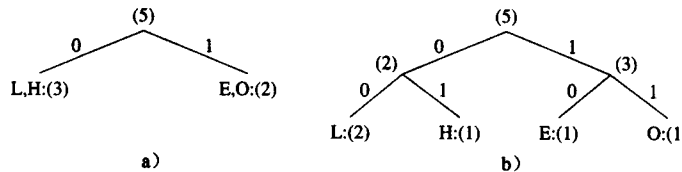


图 7-4 使用香农-凡诺算法对 HELLO 进行编码的另一种编码树

香农-凡诺算法在数据压缩时会有令人满意的编码结果，但是它很快就被赫夫曼编码方法超越并取代了。

表 7-2    在 HELLO 上应用香农-凡诺算法的另一个结果

信号	数量	$\log_2 \frac{1}{p_i}$	编码	使用的位数
L	2	1.32	00	4
H	1	2.32	01	2
E	1	2.32	10	2
O	1	2.32	11	2
总位数: 10				

7.4.2    赫夫曼编码

赫夫曼编码方法最早出现在 David A.Huffman 于 1952 年发表的论文[7]中,这种方法引发了许多深入的研究并且已被许多重要的和商业性的应用所采纳,比如传真机、JPEG 和 MPEG。

与自顶向下的香农-凡诺算法相比,赫夫曼编码的算法采用了自底向上的描述方式。仍然以单词 HELLO 为例。首先,同样要创建一棵与前面相似的二叉编码树,所有左支被编码为 0,右支被编码为 1。赫夫曼编码使用了一种简单的列表数据结构。

173

算法 7-1    赫夫曼编码

- 1) 初始化: 根据符号出现的次数对列表中所有的符号进行排序。
- 2) 重复下面步骤直到列表中只剩下一个符号。
  - ① 从列表中选出两个具有出现次数最少的符号。以这两个符号作为孩子节点创建一棵赫夫曼子树,并为这两个节点创建一个父亲节点。
  - ② 以孩子节点的出现次数的总和作为父亲节点的出现次数,将父亲节点插入字符列表中,并保持列表的有序性。
  - ③ 从列表中将孩子节点删除。
- 3) 根据从根到叶子节点的路径为每一个叶子节点赋予一个码字。

在图 7-5 中,创建了新符号 P1、P2 和 P3 来表示赫夫曼编码树中的父亲节点。列表中的内容变化如下:

- 初始化后: L H E O
- 迭代一次 (a): L P1 H
- 迭代两次 (b): L P2
- 迭代三次 (c): P3

虽然赫夫曼编码的结果通常会更好,但是对于这个简单的例子,很明显,赫夫曼编码的结果与图 7-3 中香农-凡诺算法的编码结果相同。对每个字符进行编码所需的平均位数仍然是 2 (即  $(1+1+2+3+3)/5 = 2$ )。我们再来看一个简单的例子,考虑一个包含一组字符的文本字符串,各字符出现的次数依次是: A:(15), B:(7), C:(6), D:(6)和 E:(5)。很容易知道,对这个字符串编码时,香农-凡诺算法总共需要 89 位,而赫夫曼算法只需要 87 位。

如上所示,如果可以得到准确的概率(“先验统计”),则赫夫曼编码方法会产生很好的压缩结果。只要在数据压缩前将统计数字和/编码树(在文件头中)发送给解码器,则赫夫曼编码的解码将非常简单。如果数据文件足够大的话,开销是可以忽略不计的。

赫夫曼编码具有以下几个重要性质:

- **唯一前缀性质** 任何一个赫夫曼编码都不能作为另一个赫夫曼编码的前缀。比如,图 7-5c 中分配给 L 的编码 0 不是 H 的编码 10、E 的编码 110 或是 O 的编码 111 的前缀; H 的编码

10 也不是 E 的编码 110 或 O 的编码 111 的前缀。事实上, 上面的赫夫曼算法本身保证了唯一前缀的性质, 因为它总是把所有的输入符号置于赫夫曼树的叶子节点上。赫夫曼编码是一种前缀编码, 前缀编码具有唯一的前缀。香农-凡诺算法产生的编码也是这样的一个例子。

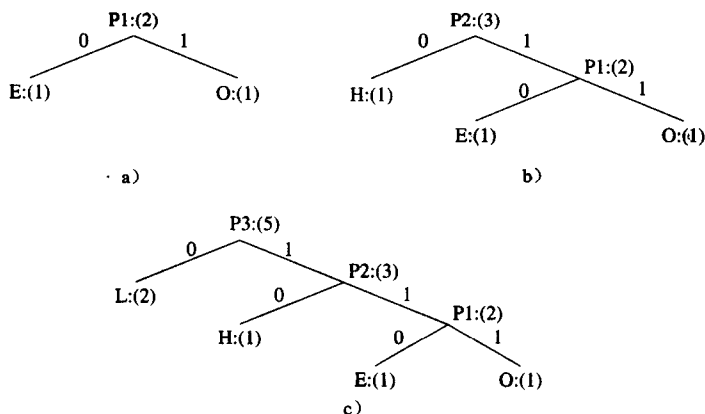


图 7-5 使用赫夫曼算法对 HELLO 进行编码的编码树

这种性质是必需的, 也有助于生成一个高效的解码器, 因为它排了解码时的多义性。在上面的例子中, 如果收到了 0, 则解码器可以立即产生符号 L 而无需等待后续的位。

- **最优性** Huffman 在 1952 年的论文[7]中写道, 赫夫曼编码是一种最小冗余编码 (minimum-redundancy code)。在[8, 2]中已经证明, 赫夫曼编码对于任何数据模型 (即一个给定的、准确的概率分布的数据模型) 都是最优的:
  - 两个频率最低的字符具有相同长度的赫夫曼编码, 但是编码的最后一位不同。从上面的算法中很容易看到这一点。
  - 出现频率较高的符号的码字比出现频率较低的符号的码字短。也就是说, 对于符号  $s_i$  和  $s_j$ , 如果  $p_i \geq p_j$ , 则  $l_i \leq l_j$ , 其中,  $l_i$  是  $s_i$  的码字长度。
  - 信息源  $S$  的平均编码长度严格小于  $\eta+1$  (参见[2])。与式 (7.5) 联立, 我们有:

$$\eta \leq \bar{l} < \eta + 1 \quad (7.6)$$

#### 扩展的赫夫曼编码

到目前为止, 我们讨论的赫夫曼编码都是给每一个符号赋予一个具有整数位长的码字。在前面我们提到,  $\log_2 \frac{1}{p_i}$  代表了包含在信息源  $s_i$  中的信息量, 它与对  $s_i$  进行编码所需的位数相等。

因而当一个符号  $s_i$  概率较大 (接近 1.0) 时,  $\log_2 \frac{1}{p_i}$  将接近于 0, 这时用 1 位来代表这个符号是相当昂贵的。只有当所有符号的概率都可以表示成  $2^{-k}$  时 ( $k$  是正整数), 码字的平均长度才可能真正地达到最优, 也就是说,  $\bar{l} \equiv \eta$ 。很显然, 大多数情况下  $\bar{l} > \eta$ 。

解决整数码字长度问题的一个方法就是将几个符号成组, 然后为整个组赋予一个码字。这种类型的赫夫曼编码称作扩展的赫夫曼编码 (extended Huffman coding) [2]。假设信息源有符号集  $S = \{s_1, s_2, \dots, s_n\}$ , 如果  $k$  个符号为一组, 那么扩展的符号集就是

$$S^{(k)} = \{\overbrace{s_1 s_1 \dots s_1}^{k \text{ 个符号}}, s_1 s_1 \dots s_1 s_2, \dots, s_1 s_1 \dots s_1 s_n, s_1 s_1 \dots s_2 s_1, \dots, s_n s_n \dots s_n\}$$

注意，新符号集合  $S^{(k)}$  的大小是  $n^k$ 。如果  $k$  相对较大（比如， $k \geq 3$ ），而且对于大多数的实际应用都有  $n \gg 1$ ，那么  $n^k$  将会是一个非常庞大的数字，这表明所需的符号表也将变得非常庞大。这个开销使得扩展的赫夫曼编码不够实用。

[2]中指出，如果  $S$  的熵是  $\eta$ ，那么  $S$  中每个符号所需的平均位数是

$$\eta \leq \bar{l} < \eta + \frac{1}{k} \tag{7.7}$$

这样我们就缩小了编码方法最好的理论 upper 界。然而，这并没有如我们所期望的那样有优于原始的赫夫曼编码（组的大小为 1）的改进。

7.4.3   自适应赫夫曼编码

赫夫曼算法需要有关信息源的先验统计知识，而这样的信息通常很难获得。这在多媒体应用中表现尤为突出，数据在到达之前是未知的，例如在直播（或流式）的音频和视频。即使能够获得这些统计数字，符号表的传输仍然是一笔相当大的开销。

对于赫夫曼编码非扩展版本，前面的讨论假定了一种 0 阶模型。也就是说，符号/字符都是相互独立的，彼此之间没有任何联系。一种包含上下文信息的方法是每次向前（或向后）考察  $k$  个符号；这就是所谓的  $k$  阶模型。例如，1 阶模型除了包含 “ $q$ ” 和 “ $u$ ” 各自的概率之外，还可以包含如 “ $qu$ ” 的概率之类的统计数字。然而，这又一次表明当  $k \geq 1$  时， $k$  阶模型需要存储和发送更多的统计数据。

解决上述问题的方案就是使用自适应压缩算法（adaptive compression algorithms），在这种算法中，统计数字是随着数据流的到达而动态地收集和更新的。概率不再是基于先验知识而是基于到目前为止实际收到的数据。之所以说这种新的编码方法是自适应的，是因为随着接收到的符号的概率分布的改变，符号将会被赋予新的（更长或更短的）码字。这种方法在内容（某一场景的音乐或色彩）和统计数字快速变化的多媒体数据中尤为适用。

作为例子，我们在这一节介绍自适应的赫夫曼编码（adaptive Huffman coding）算法。其中，许多思想也可以适用于其他的自适应压缩算法。

过程 7-1   自适应的赫夫曼编码的过程

ENCODER	DECODER
-----	-----
Initial_code();	Initial_code();
while not EOF	while not EOF
{	{
get(c);	decode(c);
encode(c);	output(c);
update_tree(c);	update_tree(c);
}	}

- Initial\_code 为符号分配一些初始的编解码双方已达成一致的码字，但是不包含任何有关频率的先验知识。比如，可以使用传统的 ACSII 码为这些字符符号编码。
- update\_tree 是一个构造自适应赫夫曼树的过程。它主要完成两个任务：将符号（包括任何新的符号）的出现次数加 1 并更新树的配置。
- 赫夫曼树必须总是保持其兄弟性质，也就是说所有的节点（内部节点或叶子节点）都是按照计数递增的顺序排列的。所有的节点按从左到右、自下而上的顺序编号。（如图 7-6 所示，第一个节点是 1.A: (1)，第二个节点是 2.B: (1)，依此类推，其中括号里的数字是出现次数。）如果违反了兄弟性质，则将触发一个交换过程对节点进行重新排列。

- 当必须进行交换时, 具有计数  $N$  的最远的节点将会与计数刚刚增加到  $N+1$  的节点交换。  
注意, 如果计数为  $N$  的节点不是叶子节点, 而是某个子树的根节点, 则该节点的整个子树将随其一同交换。

• 编码器和解码器必须使用完全相同的 Initial\_code 和 update\_tree 例程。

图 7-6a 描述了具有一些已接收到的符号的一棵赫夫曼树。图 7-6b 给出了又收到一个 A (即第二个 A) 后的更新树。这使得 A 的计数增加到  $N+1=2$  并且触发了一次交换。这时, 计数为  $N=1$  的最远的节点是 D:(1)。因此 A:(2) 和 D:(1) 交换。

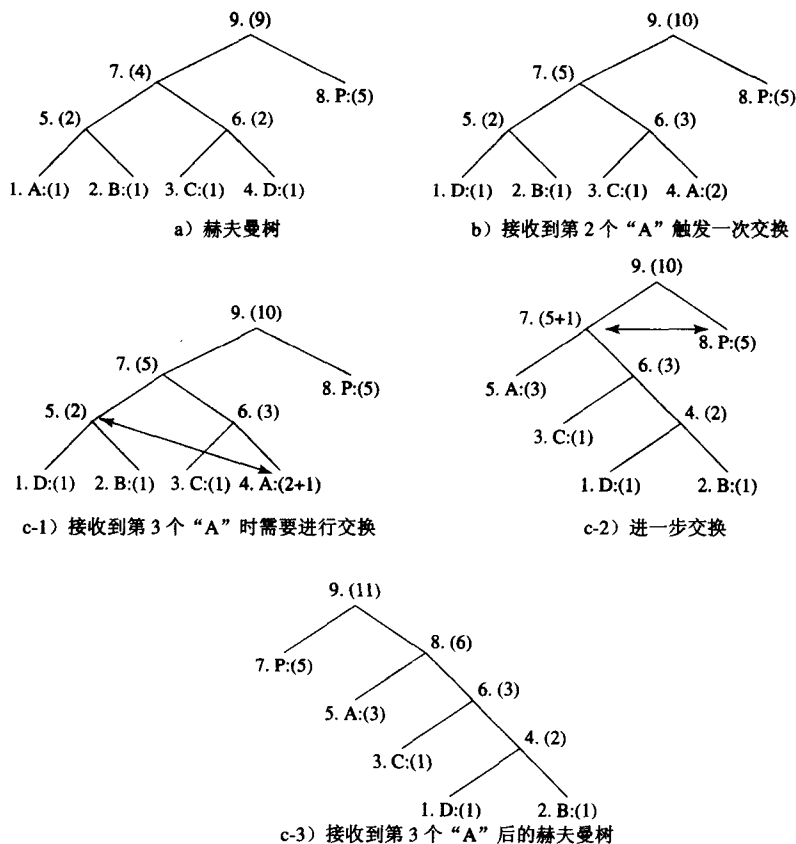


图 7-6 更新自适应赫夫曼树的节点交换过程

很明显, 先将 A:(2) 和 B:(1) 交换, 然后和 C:(1) 交换, 最后和 D:(1) 交换也可以得到同样的结果。但问题是这个过程要经过三次交换; 而“与最远的计数为  $N$  的节点”交换的原则可以帮助我们有效地避免这些不必要的步骤。

收到第 3 个 A 后赫夫曼树的更新要更复杂一些, 图 7-6c-1 到 c-3 给出了交换的三个步骤。因为 A:(2) 会变成 A:(3) (暂时记为 A:(2+1)), 所以有必要将 A:(2+1) 与第 5 个节点交换。这在图 7-6c-1 中用箭头标出。

由于第 5 个节点不是叶子节点, 所以带有节点 1.D:(1), 2.B:(1) 和 5.(2) 的子树将作为一个整体与 A:(3) 交换。图 7-6c-2 给出了第一次交换后的赫夫曼树。现在第 7 个节点的计数将变成 (5+1), 这将触发与第 8 个节点的一次交换。图 7-6c-3 给出了第二次交换后的赫夫曼树。

上面的例子给出了旨在保持自适应赫夫曼树的兄弟性质的一个更新过程——树的更新有时需

要多次交换才能完成。这时,应当采用“自底向上”的方式,从最底层需要交换的节点开始,通过多步实现交换。换句话说,更新是顺序执行的,先按顺序对树节点进行考察,在需要交换的地方交换节点。

为了清楚地说明更多的实现细节,我们来看看另一个例子。这里,我们将不再只是说明树的更新过程,而是准确的给出传送什么位。

### 例 7-1 符号串 AADCCDD 的自适应赫夫曼编码

假设给编码器和解码器分配的初始编码遵循字母表中从 A~Z26 个字母的 ASCII 码顺序,如表 7-3 所示。为了改进该算法的实现,我们采用了新的规则:在第一次发送任何字符/符号之前,必须先发送一个特殊的符号 NEW。NEW 的初始码是 0。NEW 的计数总是保持为 0 (计数永不增加);因此它总是可以表示成图 7-7 中的 NEW:(0)。

表 7-3 使用自适应赫夫曼编码为 AADCCDD 分配初始码

初始代码	
NEW:	0
A:	00001
B:	00010
C:	00011
D:	00100
⋮	⋮

图 7-7 给出了构建赫夫曼树的所有步骤。开始时,没有树。对于第一个 A,发送 0 表示 NEW,发送初始码 00001 代表 A。然后,创建树,如图中的第一棵树,标为“A”。这时,编码器和解码器都创建了相同的第一棵树,可以看出第二个 A 的编码为 1,因此发送的编码为 1。

收到第 2 个 A 后,树被更新,如图所示标为 AA。收到 D 和 C 后的更新也是类似的。这时产生了更多的子树,而 NEW 的编码也越来越长,从 0 到 00 到 000。

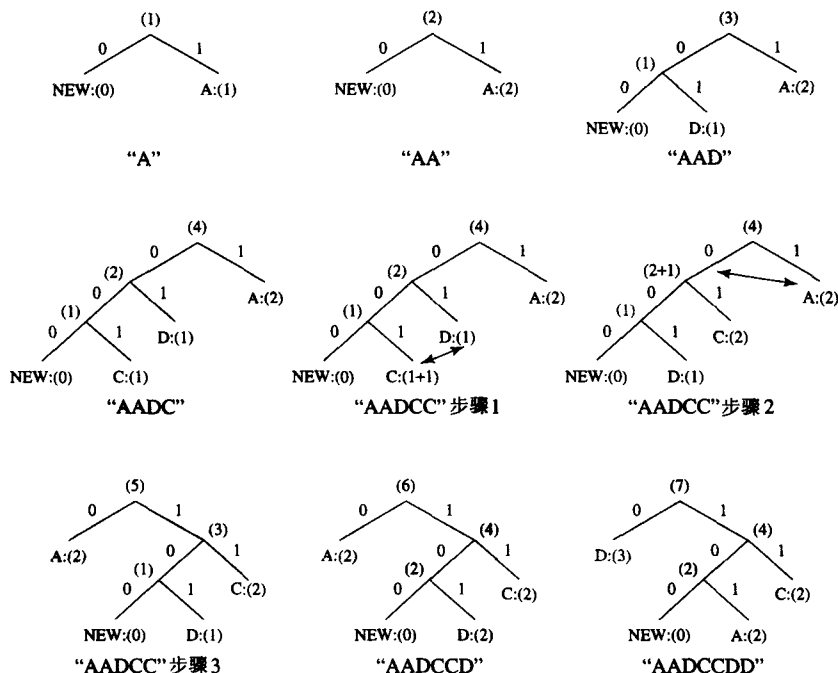


图 7-7 AADCCDD 的自适应赫夫曼树

从 AADC 到 AADCC 经过了两次交换。我们通过三个步骤来清楚地说明这个更新过程。所需

的交换仍然用箭头标出。

- **AADCC 步骤一** C 的频率计数从 1 增加到  $1+1=2$ ；这触发了 C 与 D:(1)交换。
- **AADCC 步骤二** C 和 D 交换后, C:(2)的父节点的计数将从 2 增加到  $2+1=3$ ；这又触发了 C 与 A:(2)的交换。
- **AADCC 步骤三** 最后 A 与 C 的父节点交换。

表 7-4 给出了发送给解码器的所有的符号和编码（若干 0 和 1）序列。

180

表 7-4 发送给解码器的符号和编码序列

信号	NEW	A	A	NEW	D	NEW	C	C	D	D
编码	0	00001	1	0	00100	00	00011	001	101	101

需要着重强调的是, 某个符号的编码常常会随着自适应赫夫曼编码的编码过程而改变。最近越频繁出现的符号, 编码越短。例如, 在 AADCCDD 之后, 当字母 D 取代 A 成为出现频率最高的符号时, 它的编码将从 101 变成 0。当然这也是自适应算法的基本性质, 即根据符号的新的概率分布来动态地为符号分配新的编码。

本书的网站上的“Squeeze Page”提供了自适应赫夫曼编码的 Java applet, 可以帮助你学习这个算法。

## 7.5 基于字典的编码

Lempel-Ziv-Welch (LZW) 算法利用了一种自适应的、基于字典的压缩技术。和变长编码方式（其中码字的长度不同）不同, LZW 使用定长的码字来表示通常会在一起出现（如英文中的单词）的符号/字符的变长的字符串。

和其他的自适应压缩技术一样, LZW 编码器和解码器会在接收数据时动态地创建字典, 编码器和解码器也会产生相同的字典。这样, 由于一个编码可以为一个或多个符号/字符所使用, 所以数据压缩得以实现。

LZW 在实现过程中不断将越来越长的重复条目插入字典中, 然后将字符串的编码而不是字符串本身发送出去。LZW 的前身是 Jacob Ziv 在 1977 年创建的 LZ77[9]和 Abraham Lempel 在 1978 年创建的 LZ78[10]。Terry Welch[11]在 1984 年改进了这种算法。许多应用中都采用了 LZW, 如 UNIX compress, 图像的 GIF 格式以及用于调制解调器的 V.42 bis 等。

算法 7-2 LZW 压缩

```

BEGIN
  s = next input character;
  while not EOF
  {
    c = next input character;

    if s + c exists in the dictionary
      s = s + c;
    else
    {
      output the code for s;
      add string s + c to the dictionary with a new code;
      s = c;
    }
  }
  output the code for s;
END

```

181

例 7-2    对字符串 ABABBABCABABBA 进行 LZW 压缩

我们从一个非常简单的字典（也可以称作字符串表）开始，初始时只包含三个字符和编码：

code	string
1	A
2	B
3	C

现在，如果输入字符串是 ABABBABCABABBA，则 LZW 压缩算法的工作流程如下：

s	c	output	code	string
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	A	4	10	ABA
A	B			
AB	B			
ABB	A	6	11	ABBA
A	EOF	1		

182

输出编码是 1 2 4 5 2 3 4 6 1。这样只需发送 9 个编码，而不是 14 个字符。如果我们假设每个字符或编码都用 1 个字节传送，那么所带来的节约是很可观的（这时压缩比将是  $14/9=1.56$ ）。（切记，在 LZW 这种自适应算法中，编码器和解码器都分别独立地创建自己的字符串表，因此没有任何传输字符串表的开销。）

显然，上例的输入字符串中有很多冗余，这是它能够迅速实现压缩的原因。一般来说，LZW 的节约只有在文本长度超过上百字节的时候才会显现出来。

上述的 LZW 算法很简单，它并没有选择最优的新字符串插入字典中。因此，它的字符串表会如上所示迅速增大。典型的用于文本数据的 LZW 实现使用 12 位的码长。因此，它的字典最多可以容纳 4096 个条目，其中最前面的 256（0~255）个条目是 ASCII 码。如果我们把这个考虑在内，上面的压缩比会减小到  $(14 \times 8)/(9 \times 12) = 1.04$ 。

算法 7-3    LZW 解压缩（简化版本）

```
BEGIN
s = NIL;
while not EOF
{
    k = next input code;
    entry = dictionary entry for k;
    output entry;
    if (s != NIL)
        add string s + entry[0] to dictionary
        with a new code;
    s = entry;
}
END
```

例 7-3    对字符串 ABABBABCABABBA 进行 LZW 解压缩

解码器的输入是 1 2 4 5 2 3 4 6 1。其初始字符串表与编码器的初始字符符号表一致。



LZW 解码算法的工作流程如下：

s	k	entry/output	code	string
<hr/>				
			1	A
			2	B
			3	C
<hr/>				
NIL	1	A		
A	2	B	4	AB
B	4	AB	5	BA
AB	5	BA	6	ABB
BA	2	B	7	BAB
B	3	C	8	BC
C	4	AB	9	CA
AB	6	ABB	10	ABA
ABB	1	A	11	ABBA
A	EOF			

183

显然，输出字符串是 ABABBABCABABBA——与原始字符串完全相同。

### LZW 算法细节

如果对 LZW 解压缩算法的简化版本进行仔细研究就会发现一个潜在的问题。在自适应地更新字典时，编码器的动作有时会先于解码器发生。例如，在序列 ABABB 之后，编码器将输出编码 4 并为新字符串 ABB 创建一个编码为 6 的字典条目。

解码器收到编码 4 后，产生输出 AB，此时字典中会插入一个新字符串 BA 的编码 5。这在上面的例子中会重复出现多次，比如编码器又输出了一个编码 4 和编码 6 之后。从某种意义上说，这是可以预见的。毕竟，这是一个顺序的过程，所以编码器的动作必须在解码器之前。不过在这个例子中，这并没有引发任何问题。

Welch[11]指出，LZW 解压缩算法的简化版本在下面的场景中将会出现错误。假设输入字符串是 ABABBABCABBABBAX……

### LZW 编码器：

s	c	output	code	string
<hr/>				
			1	A
			2	B
			3	C
<hr/>				
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	B			
ABB	A	6	10	ABBA
A	B			
AB	B			
ABB	A			
ABBA	X	10	11	ABBAX
		.		
		.		
		.		

184

编码器的输出编码序列（也就是解码器的输入序列）是 1 2 4 5 2 3 6 10……

简单的 LZW 解码器如下：

s	k	entry/output	code	string
-----				
			1	A
			2	B
			3	C
-----				
NIL	1	A		
A	2	B	4	AB
B	4	AB	5	BA
AB	5	BA	6	ABB
BA	2	B	7	BAB
B	3	C	8	BC
C	6	ABB	9	CA
ABB	10	???		

“???”表明解码器遇到了一个问题：找不到与最后一个输入编码 10 对应的字典条目。仔细分析会发现编码 10 是编码器端最新产生的，由字符、字符串和字符拼接而成。在本例中，字符是 A，而字符串是 BB——也就是 A+BB+A。与此同时，编码器产生的输出符号序列是 A，BB，A，BB，A。

这个例子说明，只要待编码的符号序列是字符、字符串、字符、字符串、字符，等等，编码器就会产生一个新的编码来表示字符+字符串+字符，并且在解码器还没来得及产生这个编码的时候，马上将其投入使用。

幸运的是，这是上述简化 LZW 解压缩算法唯一会出现失败的情况。而且，当这种情况发生时，变量  $s = \text{字符} + \text{字符串}$ 。这种算法的改进版本能够通过检查解码器端的字典中是否有输入编码的定义来处理这种异常。如果没有定义，它会简单地认为这个编码代表符号  $s + s[0]$ ，也就是字符+字符串+字符。

185

算法 7-4 LZW 解压缩算法（改进版本）

```
BEGIN
  s = NIL;
  while not EOF
  {
    k = next input code;
    entry = dictionary entry for k;

    /* exception handler */
    if (entry == NULL)
      entry = s + s[0];

    output entry;
    if (s != NIL)
      add string s + entry[0] to dictionary
      with a new code;
    s = entry;
  }
END
```

具体实现时字典的大小要受到实际条件的限制。比如，对于 Gif，字典最多可以有 4096 个条目，而对于 V.42 bis，字典最多可以有 2048 个条目。然而，这仍然导致了长度为 12 位或 11 位的 LZW 编码，比原始数据的字长（8 位 ASCII 码）要长。

在实际的应用中，码长  $l$  保持在  $[l_0, l_{\max}]$  范围内。对于 UNIX compress 命令来说， $l_0=9$ ， $l_{\max}$  的默认值为 16。字典初始大小为  $2^{l_0}$ 。当字典被填满时，码长将增加 1；这个过程可以重复进行，直到  $l=l_{\max}$  为止。

如果待压缩的数据没有任何可重复的结构，那么使用字典条目中新编码的可能性就很小。有时，这会导致数据膨胀（data expansion），而并非数据缩减，因为码长常常比原始数据的字长要

长。为了解决这个问题, V.42 bis 建立了两种状态: 压缩的和透明的。当检测到数据膨胀时, 压缩停止, 后一种状态被激活。

因为字典的大小有限制, 所以一旦它达到了  $2^{l_{\max}}$  个条目, LZW 将失去其自适应的能力, 而转变成静态的、基于字典的技术。UNIX compress 会在这一点上监控自己的性能。它会在压缩比低于某个阈值的时候将字典刷新并重新初始化。较好的字典管理方式应当是移除一些 LRU (最近很少使用) 的条目。V.42 bis 将会搜索字典中所有非其他条目前缀的条目, 因为这意味着这些编码在创建后并没有被使用。

186

## 7.6 算术编码

算术编码 (arithmetic coding) 是一种更现代的编码方法, 在实际中比赫夫曼编码更有效。它在 20 世纪七八十年代[12, 13, 14]已发展得比较成熟。最初的算术编码的思想是在 Shannon 1948 年的著作[3]中提出的。Peter Elias 第一次用递归方法实现了这个算法 (没有发表, 但在 Abramson 1963 的书[15]中有所提及)。这种方法后来得到了进一步的发展并在 Jelinek 1968 年[16]的书有所描述。现代算术编码要归功于 Pasco (1976) [17]、Rissanen 和 Langdon (1979) [12] 的工作。

通常情况下 (在非扩展方式中), 赫夫曼编码给每个符号分配一个整数位长度的码字。如前面所述,  $\log_2 \frac{1}{p_i}$  代表信息源  $s_i$  中所包含的信息量, 等于表示  $s_i$  所需的位数。

当某个符号  $s_i$  具有较大的概率 (接近 1.0) 时,  $\log_2 \frac{1}{p_i}$  将会接近 0, 这时用 1 位来表示这个符号是很浪费的。只有当所有符号的概率都可以表示成  $2^{-k}$  的形式时 ( $k$  是正整数), 码字的平均长度才能够到达最优, 即  $\bar{l} = \eta$  ( $\eta$  的定义如式 (7.3) 所示, 它是信息源的熵)。很明显, 多数情况下  $\bar{l} > \eta$ 。

尽管可以在分配码字时把符号划分成组 (就像在扩展的赫夫曼编码中那样) 以克服只能给每个符号分配整数位的局限, 但是赫夫曼编码器和解码器所需的结果符号表会变得非常大。

算术编码把整个消息看作一个单元。在实际中, 输入数据通常被分割成块以避免错误传播。但是, 在接下来的讨论中, 我们采用了一种非常简单的方法并引入了一个终止符。

我们用一个半开区间  $[a, b)$  表示一条消息, 其中  $a$  和  $b$  都是 0 和 1 之间的实数。初始时, 区间是  $[0, 1)$ 。随着消息长度的增加, 区间的长度将缩小, 而用来表示区间的位数将增加。假设有符号表  $[A, B, C, D, E, F, \$]$ 。其中,  $\$$  是用于表示消息结束的特殊符号, 已知的概率分布如图 7-8a 所示。

算法 7-5 算术编码的编码器

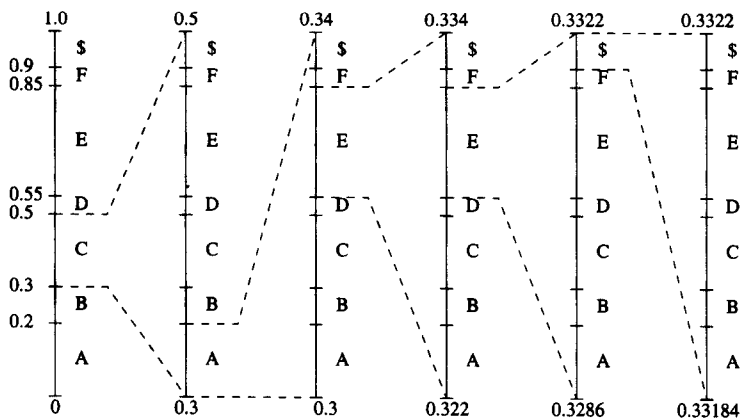
```
BEGIN
    low = 0.0;   high = 1.0;   range = 1.0;

    while (symbol != terminator)
    {
        get (symbol);
        low = low + range * Range_low(symbol);
        high = low + range * Range_high(symbol);
        range = high - low;
    }

    output a code so that low <= code < high;
END
```

信号	可能性	范围
A	0.2	[0, 0.2)
B	0.1	[0.2, 0.3)
C	0.2	[0.3, 0.5)
D	0.05	[0.5, 0.55)
E	0.3	[0.55, 0.85)
F	0.05	[0.85, 0.9)
\$	0.1	[0.9, 1.0)

a) 符号的概率分布



b) 区间缩小的图形表示

信号	低	高	范围
	0	1.0	1.0
C	0.3	0.5	0.2
A	0.30	0.34	0.04
E	0.322	0.334	0.012
E	0.3286	0.3322	0.0036
\$	0.33184	0.33220	0.00036

c) 新 low, high 和 range 值

图 7-8 算术编码: 对字符串 CAEE\$ 编码

图 7-8b 和 7-8c 给出了对字符串 CAEE\$ 进行编码的编码过程。初始时,  $low = 0$ ,  $high = 1.0$ ,  $range = 1.0$ 。在第一个符号 C 之后,  $Range\_low(C) = 0.3$ ,  $Range\_high(C) = 0.5$ ; 所以,  $low = 0 + 1.0 \times 0.3 = 0.3$ ,  $high = 0 + 1.0 \times 0.5 = 0.5$ 。这时新的 range 减小到 0.2。

为清楚起见, 图 7-8b 扩大了每一步不断减小的区间 (如虚线所示)。在第二个符号 A 之后,  $low$ ,  $high$  和  $range$  的值分别为 0.30, 0.34 和 0.04。这个过程重复进行, 直到接收到终止符 \$ 为止。这时,  $low$  和  $high$  分别是 0.33184 和 0.33220。很显然, 最后我们得到:

$$range = P_C \times P_A \times P_E \times P_E \times P_{\$} = 0.2 \times 0.2 \times 0.3 \times 0.3 \times 0.1 = 0.00036$$

编码的最后一步要求产生一个在区间  $[low, high)$  中的数字。尽管选择这样一个十进制的数字很容易, 比如 0.33184, 0.33185, 或者上例中的 0.332, 但选择一个二进制的分数就没那么明显了。下面的算法能够保证, 只要  $low$  和  $high$  分别是区间的两端并且  $low < high$ , 就能够找到一个最短的二进制码字。

## 过程 7-2 为编码器产生码字

```

BEGIN
  code = 0;
  k = 1;
  while (value(code) < low)
  {
    assign 1 to the kth binary fraction bit;
    if (value(code) > high)
      replace the kth bit by 0;
    k = k + 1;
  }
END

```

对于上面的例子,  $low = 0.33184$ ,  $high = 0.3322$ 。如果我们给第一个二进制分数位分配 1, 它的二进制值就是 0.1, 但是十进制值  $value(code) = value(0.1) = 0.5 > high$ 。因而, 我们为第一个位分配 0。因为在 while 循环进行的过程中,  $value(0.0) = 0 < low$ 。

为第二个位分配 1 会产生二进制编码 0.01 和  $value(0.01) = 0.25 < high$ , 因而可以接受。又因为  $value(0.01) < low$  也成立, 所以可以继续迭代。最终, 生成的二进制码字是 0.01010101, 也就是  $2^{-2} + 2^{-4} + 2^{-6} + 2^{-8} = 0.33203125$ 。

必须指出的是, 我们很幸运地找到了一个只有 8 位的码字来代表 CASS\$ 这个符号序列。这时,  $\log_2 \frac{1}{P_C} + \log_2 \frac{1}{P_A} + \log_2 \frac{1}{P_E} + \log_2 \frac{1}{P_E} + \log_2 \frac{1}{P_S} = \log_2 \frac{1}{range} = \log_2 \frac{1}{0.00036} \approx 11.44$ , 这表明明可以使用 12 位来对类似的符号串编码。

189

可以证明[2],  $\left\lceil \log_2 \left( \frac{1}{\prod_i P_i} \right) \right\rceil$  是上界。也就是说, 在最坏的情况下, 算术编码中最短的码字长度为  $k$  位, 并且

$$k = \left\lceil \log_2 \frac{1}{range} \right\rceil = \left\lceil \log_2 \frac{1}{\prod_i P_i} \right\rceil \quad (7.8)$$

其中,  $P_i$  是符号  $i$  的概率而  $range$  是编码器产生的最终的区间。

显然, 如果消息很长, 则区间会迅速减小, 因此,  $\log_2 \frac{1}{range}$  将变得非常大;  $\log_2 \frac{1}{range}$  和  $\left\lceil \log_2 \frac{1}{range} \right\rceil$  之间的差可以忽略不计。

一般来说, 算术编码的性能优于赫夫曼编码, 因为前者将整个消息看作一个单元, 而后者受到了必须为每个符号分配整数位的限制。例如, 赫夫曼编码需要 12 位来对 CASS\$ 编码, 而这只是算术编码中可能遇到的最坏情况。

而且, 赫夫曼编码并非总能够取得式 (7.8) 给出的上界。可以证明 (参看练习 5), 如果符号表是  $[A, B, C]$ , 已知的概率分布是  $P_A=0.5$ ,  $P_B=0.4$ ,  $P_C=0.1$ , 那么为了发送 BBB, 赫夫曼编码将需要 6 位, 比  $\left\lceil \log_2 (1/\prod_i P_i) \right\rceil = 4$  多, 而算术编码只需要 4 位。

## 算法 7-6 算术编码的解码器

```

BEGIN
  get binary code and convert to decimal value = value(code);
  Do
  {
    find a symbol s so that
      Range_low(s) <= value < Range_high(s);
    output s;
  }

```

```
        low = Rang_low(s);
        high = Range_high(s);
        range = high - low;
        value = [value - low] / range;
    }
    Until symbol s is a terminator
END
```

表 7-5 说明了上例的解码过程。初始时,  $value = 0.33203125$ 。因为  $Range\_low(C) = 0.3 \leq 0.33203125 < 0.5 = Range\_high(C)$ , 所以第一个输出符号是 C。这使得  $value = [0.33203125 - 0.3] / 0.2 = 0.16015625$ , 这个值又决定了第二个输出符号是 A。最后,  $value$  为 0.953125, 落在了终止符\$的区间[0.9,1.0)中。

190

前面描述的算法在实现上有些难度。当区间缩小时, 我们必须使用高精度的数字来编码。这就使这个算法的实际实现不太可行。幸运的是, 我们可以重新调整区间的大小并在实际实现中只使用整数算法[18]。

在上面的讨论中, 特殊符号\$是符号串的终止符。这与图像传输中发送一个 end-of-line (EOF) 很类似。传统的压缩应用不需要终止符, 因为编码器只是根据输入对所有的符号进行编码。不过, 如果传输信道/网络有噪声(有损的), 那么保留一个终止符(或者 EOL)有助于解码器与编码器保持同步。

EOL 符号本身的编码是一个有趣的问题。通常, EOL 到最后都会变得相对较长。Lei et al.[19]解决了部分问题, 并提出了一种能够控制生成的 EOL 码字长度的算法。

## 7.7 无损图像压缩

在多媒体数据压缩中应用最广泛的一种压缩技术是差分编码(differential coding)。在差分编码中, 数据减少的基础是数据流中连续符号之间存在的冗余。回想一下, 在研究如何通过预测值和误差来处理音频时, 我们在第 6 章中考虑了无损的差分编码。音频是一维的、随时间变化的信号。而这里我们期望将从音频处理中学到的知识应用到二维  $(x, y)$ , 即数字图像信号中。

### 7.7.1 图像的差分编码

我们先来研究数字图像的差分编码。从某种意义上说, 我们把一维信号及其值域转换成了二维  $(x, y)$  (图像的行与列) 上由数字索引的信号。在后面, 我们还会对视频信号进行研究。视频信号的情况更加复杂, 因为它是由空间和时间  $(x, y, t)$  索引的。

191

由于物理世界的连续性, 图像中背景和前景对象的灰度亮度(或颜色)在图像帧之间的变化是相对缓慢的。由于过去我们是在时间域上处理音频信号的, 所以实践者通常会把图像称作空间域(spatial domain)上的信号。一般来说, 变化缓慢的成像性质会使得图像空间上的相邻像素具有相似亮度值的几率变大。给定一幅原始图像  $I(x, y)$ , 使用简单的差分操作符, 我们可以按照如下方式定义一个差分图像  $d(x, y)$ :

$$d(x, y) = I(x, y) - I(x-1, y) \tag{7.9}$$

这是对由整数  $x$  和  $y$  定义的图像使用偏微分操作符  $\partial/\partial x$  的一个简单的近似。

另一种方法是利用离散的 2D Laplacian 操作符来定义差分图像  $d(x, y)$ :

$$d(x, y) = 4I(x, y) - I(x, y-1) - I(x, y+1) - I(x+1, y) - I(x-1, y) \tag{7.10}$$

表 7-5 算术编码: 对符号串 CAEE\$解码

值	输出信号	低	高	范围
0.33203125	C	0.3	0.5	0.2
0.16015625	A	0.0	0.2	0.2
0.80078125	E	0.55	0.85	0.3
0.8359375	E	0.55	0.85	0.3
0.953125	\$	0.9	1.0	0.1

在这两种情况下,图 7-9d 给出了差分图像的直方图,这个直方图是由对 7-9a 中的原始图像求偏导数  $d(x, y)$  得到的图 7-9b 中的差分图像派生的。注意,如图 7-9c 所示,  $I$  的直方图要更宽一些。可以证明,图像  $I$  的熵大于图像  $d$  的熵,因为图像  $I$  的亮度值具有更均匀的分布。因此,在差分图像上使用赫夫曼编码或其他变长编码方法会使码字的长度更短,对差分图像的压缩更有效。

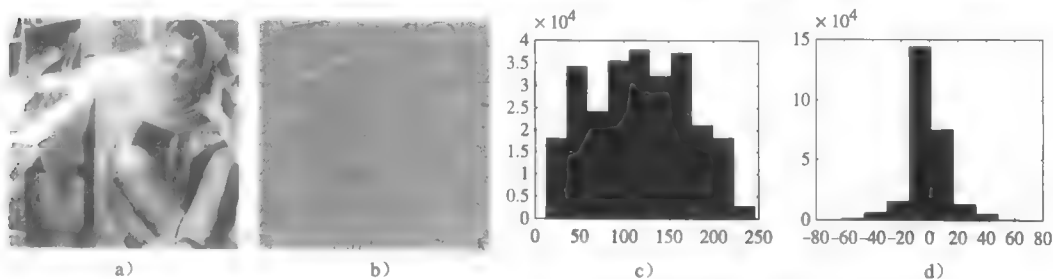


图 7-9 原始和求导图像的分布。图 a 和图 b 是原始灰度图像及其偏导数图像;

图 c 和图 d 是原始和求导图像的直方图。所用图片是通常采用的 Barb 图像

192

## 7.7.2 无损 JPEG

无损 JPEG 是 JPEG 图像压缩的一个特例。由于这种方法没有任何损失,所以它与其他 JPEG 方式有极大不同。我们把这个内容放在本章讨论,而更常用的 JPEG 方法将在第 9 章中介绍。当用户在图像工具中选择了 100% 的质量因子 (quality factor) 时,将调用无损的 JPEG 编码。为了保证完整性,无损的 JPEG 是包含在 JPEG 压缩标准中的。

下面的预测方法应用在未处理过的原始图像中 (或者原始彩色图像的每个色带中)。它主要包括两个步骤:形成差分预测和编码。

1) 预测器将相邻的三个像素值组合成当前像素的预测值,在图 7-10 中由 X 表示。预测器可以使用表 7-6 中列出的七种方案中的任何一种预测方法。如果使用预测器 P1,则相邻亮度值 A 将被用来作为当前像素的预测亮度;如果采用预测器 P4,则当前像素值就由  $A+B-C$  产生。

2) 编码器将预测值与位置 X 上的实际像素值比较并使用我们在前面讨论的某种无损的压缩算法 (如赫夫曼编码) 对两者的差值进行编码。

因为预测必须以先前已编码的相邻像素为基础,所以图像的第一个像素  $I(0, 0)$  只能使用其自身的值。第一行的像素总是使用预测器 P1,而第一列的像素总是使用预测器 P2。

无损的 JPEG 的压缩率较低,这使得它不大适用于大多数的多媒体应用。使用大约 20 幅图像得到的结果表明,无论使用何种预测器,无损 JPEG 的压缩率总在 1.0~3.0 之间浮动,平均值在 2.0 左右。考虑了水平和垂直维度上的邻近节点的预测器 4~7 比预测器 1~3 能够提供更好的压缩质量 (大约要高 0.2~0.5)。

利用测试图像 Lena、football、F-18 和 flowers,表 7-7 给出了几种无损压缩技术的压缩率之间的比较。这些标准图像可以在本书的网站上有本章的 Further Exploration 部分中找到。

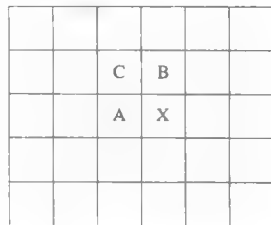


图 7-10 无损 JPEG 中预测器的相邻像素。

注意,在编码/解码环路的解码器端, A, B, C 在被预测器使用前,已经被解码

本章对各种无损压缩算法进行了探讨和研究。可以看到，这些压缩算法的压缩率通常是有较大限制的（最大值在 2~3 左右）。我们在后续几章中将要讨论的多媒体应用都需要更高的压缩率，而这只能依靠有损压缩方法来实现。

表 7-6 无损 JPEG 的预测器

预测器	预测值
P1	A
P2	B
P3	C
P4	A+B-C
P5	A+(B-C)/2
P6	B+(A-C)/2
P7	(A+B)/2

表 7-7 无损 JPEG 和其他无损压缩方法的比较

压缩方法	压缩率			
	Lena	Football	F-18	Flowers
无损 JPEG	1.45	1.54	2.29	1.26
最优无损 JPEG	1.49	1.67	2.71	1.33
压缩(LZW)	0.86	1.24	2.21	0.87
gzip (LZ77)	1.08	1.36	3.10	1.05
gzip-9(最优 LZ77)	1.08	1.36	3.13	1.05
pack(赫夫曼编码)	1.02	1.12	1.19	1.00

## 7.8 进一步探索

193  
194

和 Khalid Sayood[2]的书一样，Mark Nelson 的著作[1]也是有关数据压缩的标准参考。

本书的网站上有有关本章内容的“Further Exploration”部分提供了大量有关无损压缩的网络资源，其中包括：

- 由 Mark Nelson 编译的一套绝好的有关数据压缩的资源，包括图书馆，文献，以及赫夫曼编码、自适应赫夫曼编码、LZW 和算术编码等算法的源代码。
- 自适应算术编码的源代码。
- 数据压缩理论的网页，其中介绍了有关无损和有损数据压缩技术的基本理论。Shannon 在 1948 年发表的有关信息论的文章也可以在这个网站上下载。
- comp.compression 和 comp.compression.research 组的 FAQ。这些 FAQ 回答了数据压缩方面的很多常见问题。
- 一套能够展示自适应赫夫曼、LZW 等无损压缩交互示例的小程序。（值得一提的是，这个网页是一个本科三年级的学生在阅读了本书之后，在多媒体期末课程设计中取得的成果。）
- 有关算术编码的详尽介绍。
- 灰度测试图像 f-18.bmp, flowers.bmp, football.bmp 以及 lena.bmp。

## 7.9 练习

1. 假设 8 个字符的分布是 A:(1)、B:(1)、C:(1)、D:(2)、E:(3)、F:(5)、G:(5)、H:(10)。画出这个分布的赫夫曼树。（因为这个算法可以按照不同的顺序对具有相同概率的子树进行排列，所以答案不唯一。）
2. (a) 下面这个图像的熵 ( $\eta$ ) 是多少？其中数字 (0, 20, 50, 99) 代表灰度亮度。

99	99	99	99	99	99	99	99
20	20	20	20	20	20	20	20
0	0	0	0	0	0	0	0
0	0	50	50	50	50	0	0
0	0	50	50	50	50	0	0
0	0	50	50	50	50	0	0
0	0	50	50	50	50	0	0
0	0	0	0	0	0	0	0



- (b) 给出在对这幅图像的上述四个灰度值进行编码时构造赫夫曼树的详细步骤。写出每一个亮度值的结果编码。
- (c) 使用你的赫夫曼编码，每个像素所需的平均位数是多少？与  $\eta$  比较有什么结果？
3. 考虑一个有两个符号  $A$  和  $B$  的符号表，其概率是  $P(A) = x, P(B) = 1 - x$ 。
- (a) 绘出以  $x$  为自变量，熵为因变量的函数曲线。（ $\log_2 3 = 1.6, \log_2 7 = 2.8$ 。）
- (b) 讨论为什么在  $\epsilon$  较小时，两个符号的概率分别是  $1/2 + \epsilon$  和  $1/2 - \epsilon$  时熵会比最大值小。
- (c) 证明：对于一个能够产生  $N$  个符号的信息源，当所有符号的概率相等时，熵最大。
- (d) 作为一个小型的编程项目，编写代码验证上面的结果。
4. 扩展的赫夫曼编码为一个有  $k$  个符号的组分配一个码字。但是为什么  $average(l)$ （每个符号所需的位数）仍然不低于式 (7.7) 给出的熵  $\eta$ ？
5. (a) 与赫夫曼编码相比，算术编码有哪些优缺点？
- (b) 假设有符号表  $[A, B, C]$ ，已知概率分布是  $P_A = 0.5, P_B = 0.4, P_C = 0.1$ 。为简便起见，我们也假设编码器和解码器都认为消息的长度总是 3，因此，不需使用终止符。
- i. 计算使用赫夫曼编码对消息 BBB 进行编码所需的位数？
- ii. 计算使用算术编码对消息 BBB 进行编码所需的位数？
6. (a) 与原始的赫夫曼编码算法相比，自适应的赫夫曼编码有哪些优点？
- (b) 假设使用自适应赫夫曼编码对具有四个字母 ( $a, b, c, d$ ) 的信息源  $S$  进行编码。在进行传输之前，初始的编码是  $a=00, b=01, c=10, d=11$ 。如图 7-7 给出的例子，如果符号是第一次被发送，则必须首先发送特殊符号 NEW。
- 图 7-11 是发送了字母 aabb 之后的自适应赫夫曼树。之后，解码器接收到的后续几个字母的编码位流是 01010010101。
- i. 接收到的后续的几个字母是什么？
- ii. 画出接收每一个后续字母后的自适应赫夫曼树。
7. 比较自适应赫夫曼编码和自适应算术编码的自适应率（后者可参看本书的网站）。这些方法不能够很好地适应源统计的快速变化的原因是什么？
8. 考虑基于字典的 LZW 压缩算法。假设符号表包含符号  $\{0, 1\}$ 。给出其对应的字典（符号集合以及相关的编码）以及当输入为 0110011 时，使用 LZW 压缩算法的输出。
9. 使用你自己喜欢的编程语言编程实现赫夫曼编码、自适应赫夫曼编码、算术编码和 LZW 编码算法。自己设置至少三种不同的统计数据源来测试这些算法的实现。就每种数据源的压缩率，比较和评价各种算法的性能。

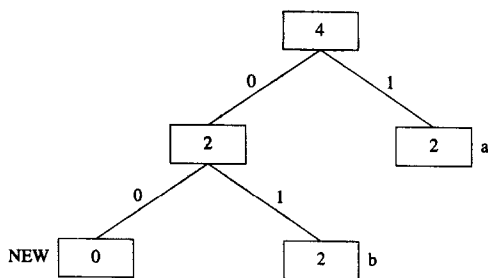


图 7-11 自适应赫夫曼树

## 7.10 参考文献

- [1] M. Nelson, *The Data Compression Book*, 2nd ed., New York: M&T Books, 1995.
- [2] K. Sayood, *Introduction to Data Compression*, 2nd ed., San Francisco: Morgan Kaufmann, 2000.
- [3] C.E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, 27: 379-423 and 623-656, 1948.
- [4] C.E. Shannon and W. Weaver, *The Mathematical Theory of Communication*, Champaign, IL:

University of Illinois Press, 1949.

- [5] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2002.
- [6] R. Fano, *Transmission of Information*, Cambridge, MA: MIT Press, 1961.
- [7] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE* [Institute of Radio Engineers, now the IEEE], 40(9): 1098–1101, 1952.
- [8] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, Cambridge, MA: MIT Press, 1992.
- [9] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, 23(3): 337–343, 1977.
- [10] J. Ziv and A. Lempel, "Compression of Individual Sequences Via Variable-Rate Coding," *IEEE Transactions on Information Theory*, 24(5): 530–536, 1978.
- [11] T.A. Welch, "A Technique for High Performance Data Compression," *IEEE Computer*, 17(6): 8–19, 1984.
- [12] J. Rissanen and G.G. Langdon, "Arithmetic Coding," *IBM Journal of Research and Development*, 23(2): 149–162, 1979.
- [13] I.H. Witten, R.M. Neal, and J.G. Cleary, "Arithmetic Coding for Data Compression," *Communications of the ACM*, 30(6): 520–540, 1987.
- [14] T.C. Bell, J.G. Cleary, and I.H. Witten, *Text Compression*, Englewood Cliffs, NJ, Prentice Hall, 1990.
- [15] N. Abramson, *Information Theory and Coding*, New York: McGraw-Hill, 1963.
- [16] F. Jelinek, *Probabilistic Information Theory*, New York: McGraw-Hill, 1968.
- [17] R. Pasco, "Source Coding Algorithms for Data Compression," Ph.D. diss., Department of Electrical Engineering, Stanford University, 1976.
- [18] P. G. Howard and J. S. Vitter, "Practical Implementation of Arithmetic Coding," *Image and Text Compression*, ed. J. A. Storer, Boston: Kluwer Academic Publishers, 1992, 85–112.
- [19] S.M. Lei and M.T. Sun, "An Entropy Coding System for Digital HDTV Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, 1(1): 147–154, 1991.

## 第 8 章 有损压缩算法

在这一章里, 我们讨论有损压缩方法。由于信息缺损意味着在误差和帧率之间权衡, 因此我们首先考虑失真 (distortion) 的量度, 如方差。接下来介绍不同的量化器, 每一种量化器都有各自不同的失真情况。对于变换编码的介绍包括用于 JPEG 压缩 (见第 9 章) 的离散余弦变换和 Karhunen Loève (K-L) 变换。此外还简单介绍了另一种变换方案——小波编码。

### 8.1 简介

第 7 章介绍过, 当图像的直方图相对平坦时, 对图像数据采用无损压缩技术 (如赫夫曼编码、算术编码和 LZW), 其压缩率 (compression ratio) 很低; 而在多媒体应用中的图像压缩需要较高的压缩率, 因而通常采用有损压缩方法。在有损压缩中, 被压缩的图像和原图像一般不完全相同, 而是得到一个感觉上 (perceptually) 与原图像的接近的近似结果。为了在数值上描述这个近似结果与原图像的接近程度, 需要采用某种形式的失真量度。

### 8.2 失真量度

失真量度 (distortion measure) 是一个说明在某种失真标准下一个近似值与原值的接近程度的数学量。对于被压缩的数据, 很自然想到失真就是原数据和重现数据在数值上的差异。然而, 如果被压缩的数据是图像, 那么这样的度量方法可能无法得到理想的结果。

举个例子, 如果重现图像与原图像除了向右移动了一条纵向扫描线的距离外完全相同, 一般很难分辨出来, 因而可以断定失真很小。然而由于重现图像每个像素的巨大改变, 通过单纯数值计算得到的结果是很大的失真。问题的关键在于我们需要的是一个感知失真 (perceptual distortion) 的量度, 而不是数值方法。不过对感知失真的研究已经超出了本书的范围。

在众多已经定义的数值化失真量度中, 本书介绍图像压缩中最常用的三种。如果关心的是像素的平均差异, 常常采用均方差 (Mean Square Error, MSE)  $\sigma^2$  量度, 其定义如下:

$$\sigma^2 = \frac{1}{N} \sum_{n=1}^N (x_n - y_n)^2 \quad (8.1) \quad \boxed{199}$$

其中  $x_n$ ,  $y_n$  和  $N$  分别为输入数据序列、重现数据序列和数据序列的长度。

如果关心的是相对于信号的误差大小, 则可以采用信噪比 (SNR) 量度。如第 6 章中所述, 信噪比就是原数据序列的均方和均方差的比值, 以分贝为单位, 其定义如下:

$$SNR = 10 \log_{10} \frac{\sigma_x^2}{\sigma_d^2} \quad (8.2)$$

其中  $\sigma_x^2$  为原数据序列的均方,  $\sigma_d^2$  为均方差。

另一个常用的失真量度是峰值信噪比 (Peak-Signal-to-Noise Ratio, PSNR), 它测量的是相对于信号峰值  $x_{\text{peak}}$  的误差大小, 其定义如下:

$$PSNR = 10 \log_{10} \frac{x_{\text{peak}}^2}{\sigma_d^2} \quad (8.3)$$

### 8.3 比率失真理论

有损压缩中永恒的问题就是比率和失真之间的权衡。比率就是重现源信号所需的平均位数，在本书中用比率失真函数  $R(D)$  来表示比率和失真之间的权衡。

直观地说，对于一个给定的源和给定的失真量度，如果  $D$  为失真容忍量，在能保证  $D$  允许的失真范围内， $R(D)$  表示源数据编码的最低比率。很显然，在  $D=0$  的情况下，就是对源数据进行无损压缩。比率失真函数可以描述对编码算法性能的基本限制，因而可用来评价不同算法的性能。

图 8-1 所示为一个典型的比率失真函数。从中可以看出， $D=0$  也就是没有缺损时的最小比率正是源数据的熵；而在比率  $R(D)=0$  时失真达到最大值，这时完全没有进行编码。

对于一个给定源，要找到一个闭合的解析式表示比率失真函数，即使可能的话，也是非常难的。Gyorgy[1]给出了不同源的比率失真函数的解析表达式。对于那些不易得到解析解的源，其率失真函数可以采用 Arimoto[2] 和 Blahut[3] 给出的算法得到数值解。

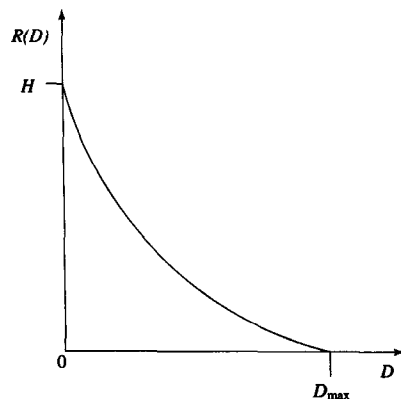


图 8-1 典型率失真函数

### 8.4 量化

对于任何有损方法，量化都是其核心。如果没有量化，只会丢失很少的信息。下面将在 6.3.2 节的基础上更详细地讨论量化。

我们所要压缩的源可能包含大量的不同的输出值（如果是模拟信号，值的数量甚至会是无穷多）。为了有效地表示源数据输出，必须通过量化把不同的值的数量大大减小。

每种算法（即每个量化器）都可以由编码器端的输入范围划分和解码器端的输出值域唯一确定。量化器的输入值和输出值可以是标量也可以是向量，这样就有了标量量化器（scalar quantizer）和矢量量化器（vector quantizer）之分。本节将分析均匀标量量化器和非均匀标量量化器的设计，并且简要介绍矢量量化（Vector Quantization, VQ）的知识。

#### 8.4.1 均匀标量量化

均匀标量量化器将输入值域划分成等距的区间，不过有时可能需要除去两边最外部的区间。区间的端点称为量化器的判定边界（decision boundaries）。每个区间对应的输出值（或者重现值）取该区间的中点值，区间的长度称作步长（step size），记为  $\Delta$ 。均匀标量量化器可以分为两种：中高型（midrise）和中宽型（midtread）。在中宽型量化器中，0 可以作为一个输出值，而中高型量化器有一个包含零的区间（见图 8-2）。中高型量化器用于输出级数为偶数的情形，中宽型量化器则用于输出级数为奇数的情形。

当源数据以很小的正数和负数之间的波动代表零值时，中宽型量化器就很有用了。在这种情况下，采用中宽型量化器就可以精确稳定地表示零值。在遇到  $\Delta=1$  的特殊情况时，量化器的输出值可以由如下公式计算：

$$Q_{\text{midrise}}(x) = \lceil x \rceil - 0.5 \quad (8.4)$$

$$Q_{\text{midtread}}(x) = \lfloor x + 0.5 \rfloor \quad (8.5)$$

一个好的均匀量化器的作用是使一个给定源输入在要求的输出值数目下的失真最小。这可以

通过调节步长 $\Delta$ 与输入数据相匹配来实现。

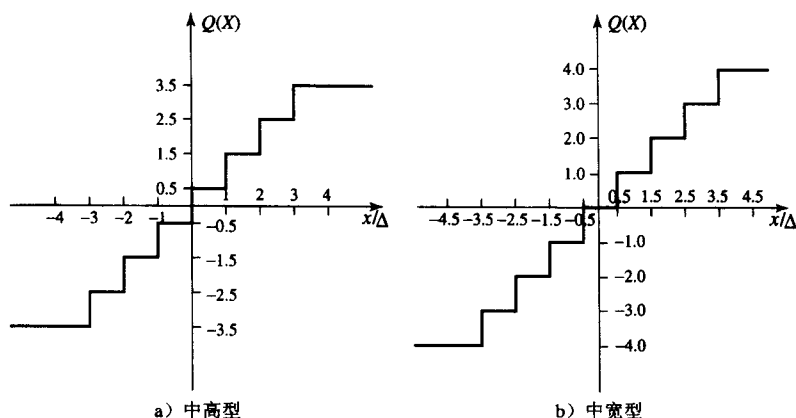


图 8-2 均匀标量量化器

考察一个  $M$  级量化器的性能。判定边界集合为  $B = \{b_0, b_1, \dots, b_M\}$ , 输出值集合为  $Y = \{y_1, y_2, \dots, y_M\}$ 。假设输入均匀分布于区间  $[-X_{\max}, X_{\max}]$  上, 则量化器的比率为

$$R = \lceil \log_2 M \rceil \quad (8.6)$$

$R$  就是对  $M$  个值进行编码所需要的位数, 具体到这个例子就是  $M$  个输出值级别。

由于输入值的范围是从  $-X_{\max}$  到  $X_{\max}$ , 因此步长 $\Delta$ 可由下式得出

$$\Delta = \frac{2X_{\max}}{M} \quad (8.7)$$

对有界限的输入, 量化器引起的量化误差称作粒度失真 (granular distortion)。如果量化器替代整个范围内的值, 从最大值到 $\infty$ , 对负值也同样如此, 这时的失真称为过载 (overload) 失真。

为了全面了解粒度失真, 我们注意到中高型量化器的判别边界  $b_i$  为  $[(i-1)\Delta, i\Delta]$  ( $i=1, \dots, M/2$ ), 这里只考虑数据  $X$  为正值的情况 ( $X$  为负值的情况对应另一半判定边界)。输出值  $y_i$  为对应区间的中点  $i\Delta - \Delta/2$  ( $i=1, \dots, M/2$ ), 这里仍然只考虑数据为正值的情况。这样总失真就是数据为正值时失真总和的两倍, 如下式所示:

$$D_{\text{gran}} = 2 \sum_{i=1}^{\frac{M}{2}} \int_{(i-1)\Delta}^{i\Delta} \left( x - \frac{2i-1}{2} \Delta \right)^2 \frac{1}{2X_{\max}} dx \quad (8.8)$$

式中将结果除以  $X$  的范围以进行归一化处理。

由于重现值  $y_i$  为每个区间的中点, 所以量化误差必然落在区间  $\left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right]$  内。图 8-3 为一个均匀分布源的量化误差图。

在这种情况下, 量化误差也是均匀分布的。因此其均方差与由误差值在  $\left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right]$  内的区间  $[0, \Delta]$  计算出来的量化误差的

方差  $\sigma_d^2$  是相同的。在点  $x$  处的误差值为  $e(x) = x - \Delta/2$ , 所以误差的方差如下:

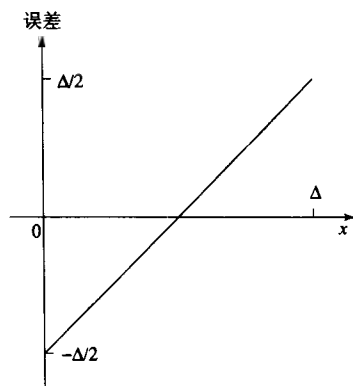


图 8-3 一个均匀分布源的量化误差

$$\begin{aligned}
 \sigma_d^2 &= \frac{1}{\Delta} \int_0^{\Delta} (e(x) - \bar{e})^2 dx \\
 &= \frac{1}{\Delta} \int_0^{\Delta} \left( x - \frac{\Delta}{2} - 0 \right)^2 dx \\
 &= \frac{\Delta^2}{12}
 \end{aligned} \tag{8.9}$$

203

类似地可以得出信号的方差为  $\sigma_x^2 = (2X_{\max})^2 / 12$ ，这样如果量化器是  $n$  位的， $M = 2^n$ ，那么从式 (8.2) 可以得到

$$\begin{aligned}
 SQNR &= 10 \log_{10} \left( \frac{\sigma_x^2}{\sigma_d^2} \right) \\
 &= 10 \log_{10} \left( \frac{(2X_{\max})^2}{12} \cdot \frac{12}{\Delta^2} \right) \\
 &= 10 \log_{10} \left( \frac{(2X_{\max})^2}{12} \cdot \frac{12}{\left( \frac{2X_{\max}}{M} \right)^2} \right) \\
 &= 10 \log_{10} M^2 = 20n \log_{10} 2
 \end{aligned} \tag{8.10}$$

$$= 6.02n(\text{dB}) \tag{8.11}$$

因此我们重新推导出了式 (6.3)，这个公式在 6.1 节中的推导是简化的。从式 (8.11) 可以得到一个重要的结论：量化器每增加 1 位信号，量化噪声比就增加 6.02dB。对  $D$  更精确的估计必须采用更精确的误差的概率分布的模型。

### 8.4.2 非均匀标量量化

如果输入源不是均匀分布的，均匀量化器就可能失去作用。在源密集分布的区域增加判定级数量可以有效地降低细粒失真。另外，还可以扩大源稀疏分布的区域而不必增加总的判定级数量。这样的非均匀量化器 (nonuniform quantizer) 就有非均匀定义的判定边界。

非均匀量化有两种常用的方法：Lloyd-Max 量化器和压缩扩展量化器，这两种方法在第 6 章中都有介绍。

#### 1. Lloyd-Max 量化器\*

对均匀量化器来说，总失真就等于粒度失真，如式 (8.8) 所示。如果源不是均匀分布的，就必须考虑其概率分布 (概率分布函数)  $f_X(x)$ 。现在我们需要同时求解以修正判定边界  $b_i$  和重现值  $y_i$ 。为此将变量  $b_i$  和  $y_i$  插入总失真量度

$$D_{\text{gran}} = \sum_{j=1}^M \int_{b_{j-1}}^{b_j} (x - y_j)^2 \frac{1}{X_{\max}} f_X(x) dx \tag{8.12}$$

接下来设式 (8.12) 的导数为零以使总失真最小，对  $y_i$  求导就得到重现值集合

$$y_j = \frac{\int_{b_{j-1}}^{b_j} x f_X(x) dx}{\int_{b_{j-1}}^{b_j} f_X(x) dx} \tag{8.13}$$

204

这就意味着最优的重现值就是  $x$  区间的加权中心。

对  $b_j$  求导并令其等于零可以得出：

$$b_j = \frac{y_{j+1} + y_j}{2} \quad (8.14)$$

这样得到的判定边界  $b_j$  是两个相邻重现值的中点。

通过迭代法同时求解这两个方程，其结果就是 Lloyd-Max 量化器。

#### 算法 8-1 Lloyd-Max 量化器

```

BEGIN
  Choose initial level set  $y_0$ 
   $i = 0$ 
  Repeat
    Compute  $b_i$  using Equation 8.14
     $i = i + 1$ 
    Compute  $y_i$  using Equation 8.13
  Until  $|y_i - y_{i-1}| < \epsilon$ 
END

```

从最优重现级别的初始假设开始，上述算法由当前重现级别的估计值迭代估算出最优边界，然后采用刚刚计算出的边界信息更新当前重现级别的估计值。这个过程重复进行，直到重现级别收敛为止。该算法实际的例子参见练习 3。

#### 2. 压缩扩展量化器

在压缩扩展量化中，输入通过一个压缩函数  $G$  映射后由一个均匀量化器进行量化。经过变换后再用扩展函数  $G^{-1}$  将量化后的值映射回去。图 8-4 所示为压缩扩展过程的示意图，其中  $\hat{X}$  是  $X$  的量化值。对于有上界  $X_{\max}$  的输入源，任何非均匀量化器都可以由压缩扩展量化器来表示。常用的两个压缩扩展器（comparer）是  $\mu$  律压缩扩展器和 A 律压缩扩展器（参见 6.1 节）。

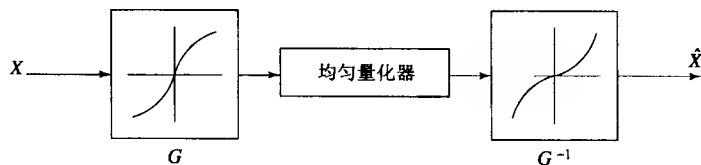


图 8-4 压缩扩展量化

#### \*8.4.3 矢量量化

香农关于信息理论的著作中最基本的思想，就是任何压缩系统对矢量或成组样本进行操作的效果要比其对单独的信号或样本进行操作的效果好。将一系列连续的样本连接成一个向量，这样就可以构造输入样本向量。例如，输入向量可以是一次讲话的片段，一幅图片中一组连续的像素或者任何格式的一块数据。

矢量量化（VQ）的思想和标量量化的思想类似，只不过将其扩展到了多个维度。在标量量化中，用一个重现值来表示一维空间的一段区间；而在矢量量化中用一个包含  $n$  个分量的码向量（code vector）来表示  $n$  维空间某个区域内的向量。这些码向量的集合构成了向量量化器中的码本（codebook）。

与一维情形不同，码向量没有固定的排列顺序，因而需要一个指针集来对码本进行索引。图 8-5 给出了向量量化的基本步骤。在图中，编码器找出与输入向量最接近的码向量后输出对应的指针。在解码器端用的是完全相同的码本。当接收到输入向量的编码指针后就可以通过简单的表查询来确定重现向量。

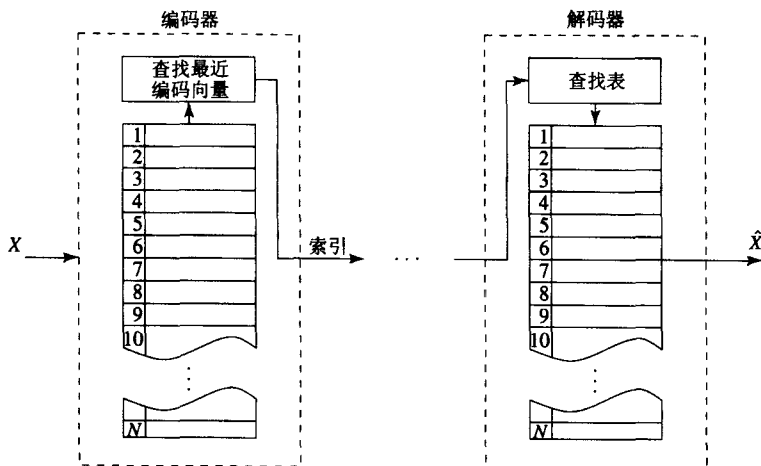


图 8-5 向量量化的基本过程

206

得到合适的码本和在编码器端搜索最接近的码向量这两项工作需要相当多的计算资源。不过解码器可以很快完成，因为完成重现只需一个固定时间的操作。由于有这样的特性，VQ 非常适合在那些编码器端有大量资源而解码器端只有有限资源，并且需要很短操作时间的系统上使用。大多数多媒体的应用都采用这种类型。

## 8.5 变换编码

从信息理论的基本原理可知，矢量编码比标量编码效率更高（参见 7.4.2 节）。为了实现这个目的，我们需要将源输入中的连续样本块聚合成矢量。

设  $X = \{x_1, x_2, \dots, x_k\}^T$  为样本向量。不论输入数据是一张图片、一段音乐还是一段音频或视频剪辑甚至是一段文本，相邻样本  $x_i$  之间都可能有着密切的内在相关性。变换编码的基本原理是：如果  $Y$  是对输入向量  $X$  进行线性变换  $T$  的结果，线性变换  $T$  使得  $Y$  的元素间的相关性比  $X$  中元素的相关性更弱，那么对  $Y$  的编码效率就比对  $X$  编码的效率更高。

例如，如果一张 RGB 图片的大部分信息都包含在一条主轴线上，旋转后使得轴线方向是第一个元素，这样亮度就可以采取与颜色信息不同的压缩，这样做能够更接近人眼的亮度信道。

对于大于三维的情况，如果大部分信息能够用变换后的向量中的前几个分量精确地描述，那么对剩余的分量可以只进行粗粒度的量化甚至将其设为零，而只有很小的信号失真。非相关性越大，即某一维对其他维的影响越小（正交轴越多），就越有可能对存储信息较少的轴作不同的处理，同时对量化后或截距变换后信号重现精确性的影响很小。

一般而言，变换  $T$  不对数据进行压缩，压缩是由对  $Y$  的分量的量化和处理完成的。本节将讨论解除输入信息相关性的工具：离散余弦变换 (DCT)，另外还将分析 Karhunen-Loève 变换 (KLT)，这是一种解除输入  $X$  的相关性的最优方法。

### 8.5.1 离散余弦变换

离散余弦变换 (DCT) 是一种广泛应用的变换编码方法，它能够以数据无关的方式解除输入信号之间的相关性。因此，它应用得较多。下面我们将分析 DCT 的定义并且讨论它的一些特性，特别是 DCT 与常见的离散傅里叶变换 (DFT) 的关系。



### 1. DCT 的定义

以二维 DCT 为例, 设一个有两个整数变量  $i$  和  $j$  的函数  $f(i, j)$  (图片中的一块)。二维 DCT 将其变换成一个新的函数  $F(u, v)$ , 其中整数  $u$  和  $v$  的取值范围与  $i$  和  $j$  相同, 该变换的一般定义如下:

$$F(u, v) = \frac{2C(u)C(v)}{\sqrt{MN}} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \cos \frac{(2i+1)u\pi}{2M} \cos \frac{(2j+1)v\pi}{2N} f(i, j) \quad (8.15) \quad \boxed{207}$$

其中  $i, u = 0, 1, \dots, M-1$ ;  $j, v = 0, 1, \dots, N-1$ , 常数  $C(u)$  和  $C(v)$  由下式得出:

$$C(\xi) = \begin{cases} \frac{\sqrt{2}}{2} & \xi = 0 \\ 1 & \text{其他} \end{cases} \quad (8.16)$$

在 JPEG 图像压缩标准 (参见第 9 章) 中, 一个图像块的维度定义为  $M=N=8$ 。因此, 在这种情况下, 二维 DCT 及其逆变换 (IDCT) 的定义如下:

#### • 二维离散余弦变换 (2D DCT)

$$F(u, v) = \frac{C(u)C(v)}{4} \sum_{i=0}^7 \sum_{j=0}^7 \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16} f(i, j) \quad (8.17)$$

其中  $i, j, u, v = 0, 1, \dots, 7$ , 常数  $C(u)$  和  $C(v)$  由式 (8.16) 得出。

#### • 二维逆离散余弦变换 (2D IDCT)

2D IDCT 和 2D DCT 几乎完全相同, 只是将  $f(i, j)$  和  $F(u, v)$  颠倒一下, 另外  $C(u)$  和  $C(v)$  出现在  $\Sigma$  号里面:

$$\tilde{f}(i, j) = \sum_{u=0}^7 \sum_{v=0}^7 \frac{C(u)C(v)}{4} \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16} F(u, v) \quad (8.18)$$

其中  $i, j, u, v = 0, 1, \dots, 7$ 。

二维变换适用于二维信号, 如数字图像等。如下所示, 一维 DCT 和 IDCT 和二维 DCT 和 IDCT 类似。

#### • 一维离散余弦变换 (1D DCT)

$$F(u) = \frac{C(u)}{2} \sum_{i=0}^7 \cos \frac{(2i+1)u\pi}{16} f(i) \quad (8.19)$$

其中  $i = 0, 1, \dots, 7; u = 0, 1, \dots, 7$ 。

#### • 一维逆离散余弦变换 (1D IDCT)

$$\tilde{f}(i) = \sum_{u=0}^7 \frac{C(u)}{2} \cos \frac{(2i+1)u\pi}{16} F(u) \quad (8.20)$$

其中  $i = 0, 1, \dots, 7; u = 0, 1, \dots, 7$ 。

### 2. 一维 DCT

下面分析一维信号的离散余弦变换, 这里介绍的大部分概念都可以轻松应用到二维 DCT。

有固定幅值的电信号称为直流 (Direct Current, DC) 信号, 常见的例子有电池带有 1.5V 或者 9V 的直流电。幅值以某种频率周期性变化的电信号称为交流 (Alternating Current, AC) 信号。例如, 家用电源是 60Hz、110V 的正弦波形交流电 (在其他许多国家是 50Hz、220V)。

然而, 大多数实际的信号却更加复杂。语言信号或数字图像中某一行的灰度亮度就是这种复

杂的一维信号。不过,任何信号都可以表示多个为不同振幅和频率的正弦波或余弦波信号的叠加,这一过程就叫作傅里叶分析。电气工程中的 DC 和 AC 等名词被引用过来描述信号中的分量,一个信号通常由一个 DC 分量和多个 AC 分量组成。

如果采用的是余弦函数,那么确定信号 DC 分量和 AC 分量的振幅的过程就称作余弦变换(Cosine Transform)。如果指数为整数,就是离散余弦变换。当  $u=0$  时,式(8.19)得出的是 DC 分量的系数,当  $u=1, 2, \dots, 7$  时,依次得到各个 AC 分量的系数。

式(8.20)所示为逆离散余弦变换,它使用 DC、AC 分量的系数和余弦函数来重现(重组)函数  $f(i)$ ,由于 DCT 和 IDCT 的计算过程中会产生一些信号损耗,这时的  $f(i)$  表示为  $\tilde{f}(i)$ 。

简单地说,DCT 的作用就是将原信号分解成 DC 分量和 AC 分量,IDCT 则重现(重组)信号。DCT 和 IDCT 都使用相同的余弦函数集合,这些余弦函数称作基函数(basis function)。图 8-6 所示为一维 DCT 在  $u=0,1,\dots,7$  时的八个基函数。

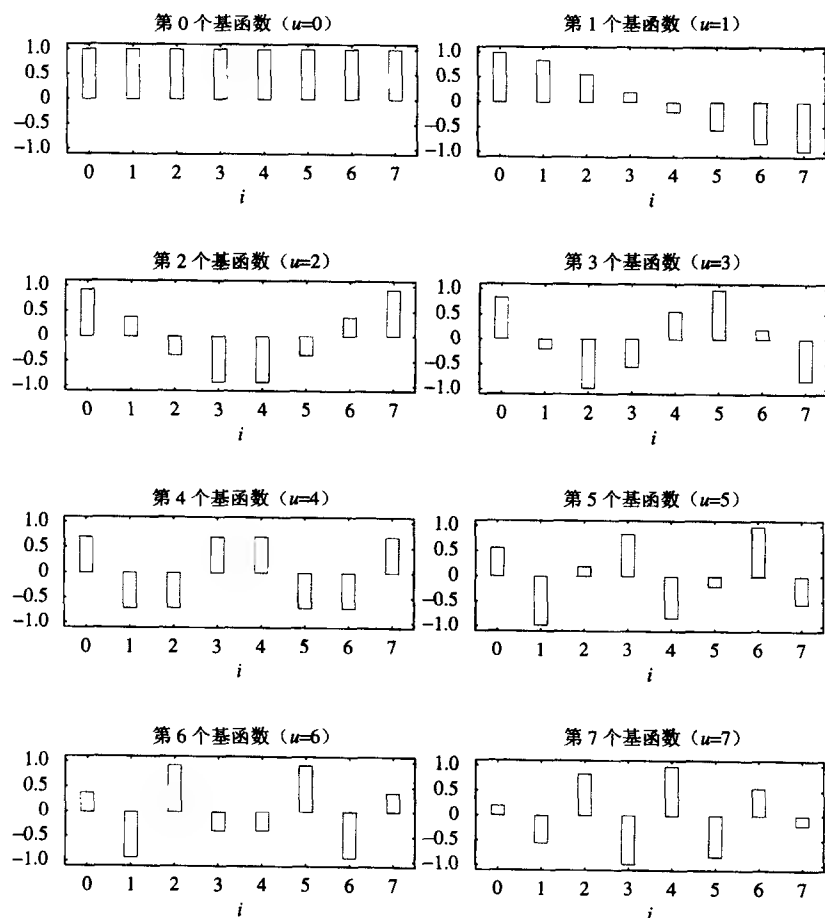


图 8-6 一维 DCT 的基函数

DCT 提供了一种新的在频域 (frequency domain) 上处理和分析信号方法,我们已经能够分析一个  $8 \times 8$  像素组成的图像块。但是我们下面要考虑时间相关的信号而不是空间相关的信号 (因为时间信号的分析正是这种方法的起源)。

设  $f(i)$  表示一个随时间  $i$  变化的信号（这里时间没有按惯例用  $t$  来表示），一维 DCT 将时域（time domain）上的函数  $f(i)$  变换为频域上的函数  $F(u)$ 。 $F(u)$  的系数称为频率响应（frequency response），由之还可以得到  $f(i)$  的频谱图。

下面用几个例子来说明频率响应。

### 例 8-1

图 8-7a 左边部分所示为一个幅值为 100 的 DC 信号，即  $f_1(i) = 100$ 。由于我们分析的是离散余弦变换，因此输入信号是离散的，其定义域为  $[0, 7]$ 。

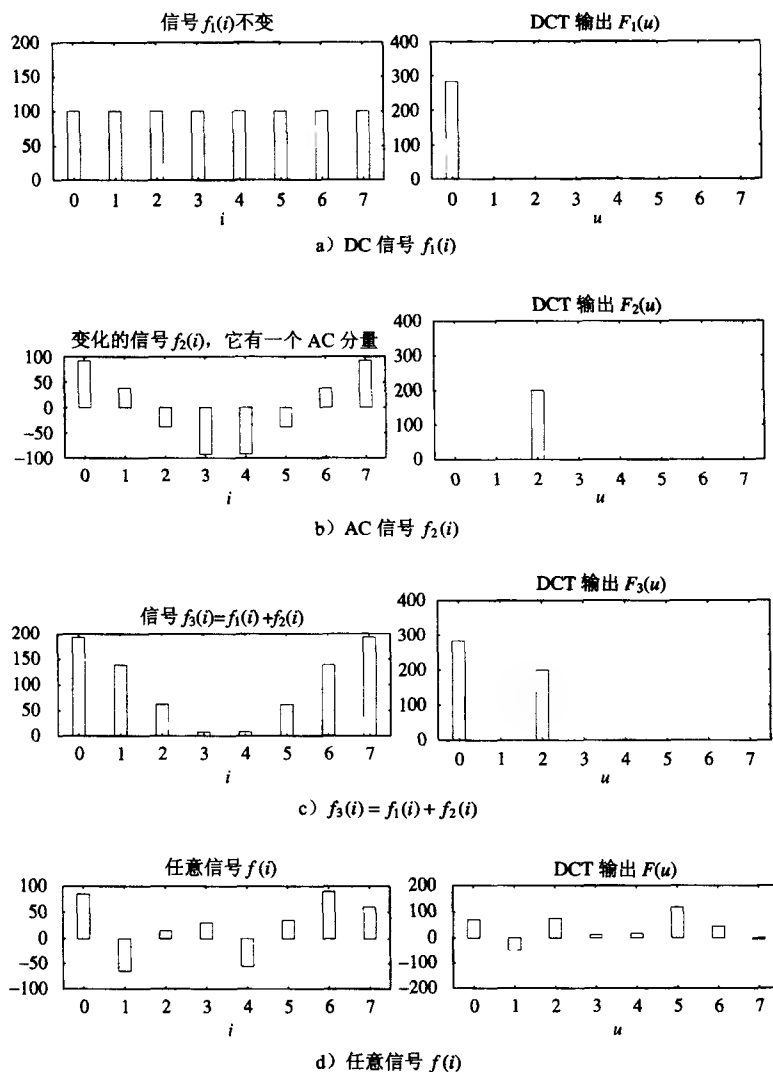


图 8-7 一维离散余弦变换的例子

当  $u=0$  时，不论  $i$  取什么值，式 (8.19) 中所有的余弦项都变成  $\cos 0$ ，也就是 1，这时  $C(0) = \sqrt{2}/2$ ， $F_1(0)$  由下式得出：

$$F_1(0) = \frac{\sqrt{2}}{2 \cdot 2} \cdot (1 \cdot 100 + 1 \cdot 100 + 1 \cdot 100 + 1 \cdot 100 \\ + 1 \cdot 100 + 1 \cdot 100 + 1 \cdot 100 + 1 \cdot 100) \\ \approx 283$$

当  $u=1$  时,  $F_1(u)$  如下所示, 因为  $\cos \frac{\pi}{16} = -\cos \frac{15\pi}{16}$ ,  $\cos \frac{3\pi}{16} = -\cos \frac{13\pi}{16}$  ..., 且  $C(1)=1$ , 所以有

$$F_1(1) = \frac{1}{2} \cdot \left( \cos \frac{\pi}{16} \cdot 100 + \cos \frac{3\pi}{16} \cdot 100 + \cos \frac{5\pi}{16} \cdot 100 + \cos \frac{7\pi}{16} \cdot 100 \right. \\ \left. + \cos \frac{9\pi}{16} \cdot 100 + \cos \frac{11\pi}{16} \cdot 100 + \cos \frac{13\pi}{16} \cdot 100 + \cos \frac{15\pi}{16} \cdot 100 \right) \\ = 0$$

同样, 可以得到  $F_1(2) = F_1(3) = \dots = F_1(7) = 0$ 。DC 信号  $f_1(i)$  的一维 DCT 结果  $F_1(u)$  如图 8-7a 右边部分所示, 变换结果  $F$  只有 DC 分量 (即第一个分量) 不为零。

### 例 8-2

图 8-7b 左边部分所示是一个离散的余弦信号  $f_2(i)$ 。它恰好与第二个余弦基函数有相同的频率和相位, 其幅值为 100。

当  $u=0$  时, 式 (8.19) 中所有的余弦项都为 1。因为  $\cos \frac{\pi}{8} = -\cos \frac{7\pi}{8}$ ,  $\cos \frac{3\pi}{8} = -\cos \frac{5\pi}{8}$  等, 所以有

$$F_2(0) = \frac{\sqrt{2}}{2 \cdot 2} \cdot 1 \cdot \left( 100 \cos \frac{\pi}{8} + 100 \cos \frac{3\pi}{8} + 100 \cos \frac{5\pi}{8} + 100 \cos \frac{7\pi}{8} \right. \\ \left. + 100 \cos \frac{9\pi}{8} + 100 \cos \frac{11\pi}{8} + 100 \cos \frac{13\pi}{8} + 100 \cos \frac{15\pi}{8} \right) \\ = 0$$

为了计算  $F_2(u)$ , 首先注意到当  $u=2$  时, 由于  $\cos \frac{3\pi}{8} = \sin \frac{\pi}{8}$ , 故有

$$\cos^2 \frac{\pi}{8} + \cos^2 \frac{3\pi}{8} = \cos^2 \frac{\pi}{8} + \sin^2 \frac{\pi}{8} = 1$$

同样

$$\cos^2 \frac{5\pi}{8} + \cos^2 \frac{7\pi}{8} = 1 \\ \cos^2 \frac{9\pi}{8} + \cos^2 \frac{11\pi}{8} = 1 \\ \cos^2 \frac{13\pi}{8} + \cos^2 \frac{15\pi}{8} = 1$$

可以得到

$$F_2(2) = \frac{1}{2} \cdot \left( \cos \frac{\pi}{8} \cdot \cos \frac{\pi}{8} + \cos \frac{3\pi}{8} \cdot \cos \frac{3\pi}{8} + \cos \frac{5\pi}{8} \cdot \cos \frac{5\pi}{8} \right. \\ \left. + \cos \frac{7\pi}{8} \cdot \cos \frac{7\pi}{8} + \cos \frac{9\pi}{8} \cdot \cos \frac{9\pi}{8} + \cos \frac{11\pi}{8} \cdot \cos \frac{11\pi}{8} \right)$$

$$\begin{aligned}
 & + \cos \frac{13\pi}{8} \cdot \cos \frac{13\pi}{8} + \cos \frac{15\pi}{8} \cdot \cos \frac{15\pi}{8} \Big) \cdot 100 \\
 & = \frac{1}{2} \cdot (1+1+1+1) \cdot 100 = 200
 \end{aligned}$$

其他项的推导过程这里不详细展开，最终得到  $F_2(1) = F_2(3) = F_2(4) = \dots = F_2(7) = 0$ 。

212

### 例 8-3

图 8-7 的第三行所示的 DCT 的输入信号是前面两个信号的叠加信号，也就是  $f_3(i) = f_1(i) + f_2(i)$ ，输出  $F(u)$  的值如下所示：

$$\begin{aligned}
 F_3(0) &= 283 \\
 F_3(2) &= 200 \\
 F_3(1) &= F_3(3) = F_3(4) = \dots = F_3(7) = 0
 \end{aligned}$$

由以上结果得出  $F_3(u) = F_1(u) + F_2(u)$ 。

### 例 8-4

图 8-7 的第四行所示是一个任意输入（或者说至少是相对复杂）的输入信号  $f(i)$  及其 DCT 输出  $F(u)$ ：

$$\begin{aligned}
 f(i)(i=0..7): & \quad 85 \quad -65 \quad 15 \quad 30 \quad -56 \quad 35 \quad 90 \quad 60 \\
 F(u)(u=0..7): & \quad 69 \quad -49 \quad 74 \quad 11 \quad 16 \quad 117 \quad 44 \quad -5
 \end{aligned}$$

在这种更一般的情形下，所有的 DCT 系数  $F(u)$  都非零，有些是负值。

根据以上的例子，可以总结出 DCT 的特性如下：

1) DCT 产生和空间信号  $f(i)$  对应的频谱  $F(u)$ 。

具体地讲，DCT 系数  $F(0)$  就是信号  $f(i)$  的 DC 分量， $F(0)$  等于信号的平均值乘以上一个常数（对于一维 DCT 是  $\frac{1}{2} \cdot \frac{\sqrt{2}}{2} \cdot 8 = 2 \cdot \sqrt{2}$ ，对于二维 DCT 是  $\frac{1}{4} \cdot \frac{\sqrt{2}}{2} \cdot \frac{\sqrt{2}}{2} \cdot 64 = 8$ ）。在图 8-7a 中，DC 信号的平均幅值显然是 100， $F(0) = 2\sqrt{2} \times 100$ ；在图 8-7b 中，AC 信号的平均幅值为 0，因而  $F(0)$  也为 0；图 8-7c 中  $f_3(i)$  的平均幅值是 100，同样  $F(0) = 2\sqrt{2} \times 100$ 。

另外 7 个 DCT 系数反映了信号  $f(i)$  在不同频率上的不同变化（即 AC）分量。如果用 AC1 表示  $F(1)$ ，AC2 表示  $F(2)$ ，…AC7 表示  $F(7)$ ，那么 AC1 就是第一个 AC 分量，其余弦函数在  $[0, 7]$  只有半个周期；AC2 有一个周期；AC3 有一个半周期；…AC7 有三个半周期。当然，所有这些都对应于以完全相同方式排列的余弦基函数，也就是说，第二个基函数对应于 AC1，第三个基函数对应于 AC2，依此类推。在图 8-7b 的例子中，由于信号  $f_2(i)$  与第三个基函数有完全相同的余弦波形，且频率和相位相等，因此它们将同时达到最大值（正）和最小值（负）。结果，它们的乘积永远是正的，它们作用的结果（ $F_2(2)$  或者 AC2）是很大的。而由于  $f_2(i)$  正好与其他基函数正交，因此其他所有的 AC 系数都为零。（关于正交性将在本章后面讨论。）

213

应该指出的是，DCT 系数很容易取负值。对于 DC 信号，当  $f(i)$  的平均值小于零时，DCT 系数就会取负值。（对于一张图片，这种情形不会发生，因而其 DC 分量是非负的。）对于 AC 信号，有一种特殊情况，如果  $f(i)$  和某一基函数的频率相同，但却相差半个周期时，系数为负，而且绝对值可能还很大。一般而言，信号更像是图 8-7d 中所示那样， $f(i)$  产生许多非零的 AC 分量，

其中 AC7 后面的表示的是高频分量。只有当信号在  $[0, 7]$  这个小区域内有迅速的变化时, 其高频分量才会有较大的 (正的或负的) 响应。

举个例子, 如果 AC7 是个很大的正数, 这就说明信号  $f(i)$  有一个与第八个基函数 (三个半周期) 同步变化的分量。根据奈奎斯特定理, 这是采用 8 个离散值取样而不发生严重失真和信号叠加的最高信号频率。

2) DCT 是一种线性变换 (linear transform)。

通常, 如果一个变换  $\mathcal{T}$  满足下面的性质, 它就是线性变换

$$\mathcal{T}(\alpha p + \beta q) = \alpha \mathcal{T}(p) + \beta \mathcal{T}(q) \quad (8.21)$$

其中  $\alpha$  和  $\beta$  是常数,  $p$  和  $q$  是任意函数、变量或者常数。

由式 (8.19) 的定义, 可以很容易证明 DCT 的这个性质, 因为 DCT 中只使用了简单的算术运算。

### 3. 一维逆 DCT

下面给出图 8-7d 中例子的逆 DCT (IDCT) 来结束这个例子。前面得到  $F(u)$  如下:

$$F(u)(u=0..7): 69 \quad -49 \quad 74 \quad 11 \quad 16 \quad 117 \quad 44 \quad -5$$

式 (8.20) 给出的一维 IDCT 能够很容易地以一个 8 次迭代的循环来实现, 这一过程如图 8-8 所示。

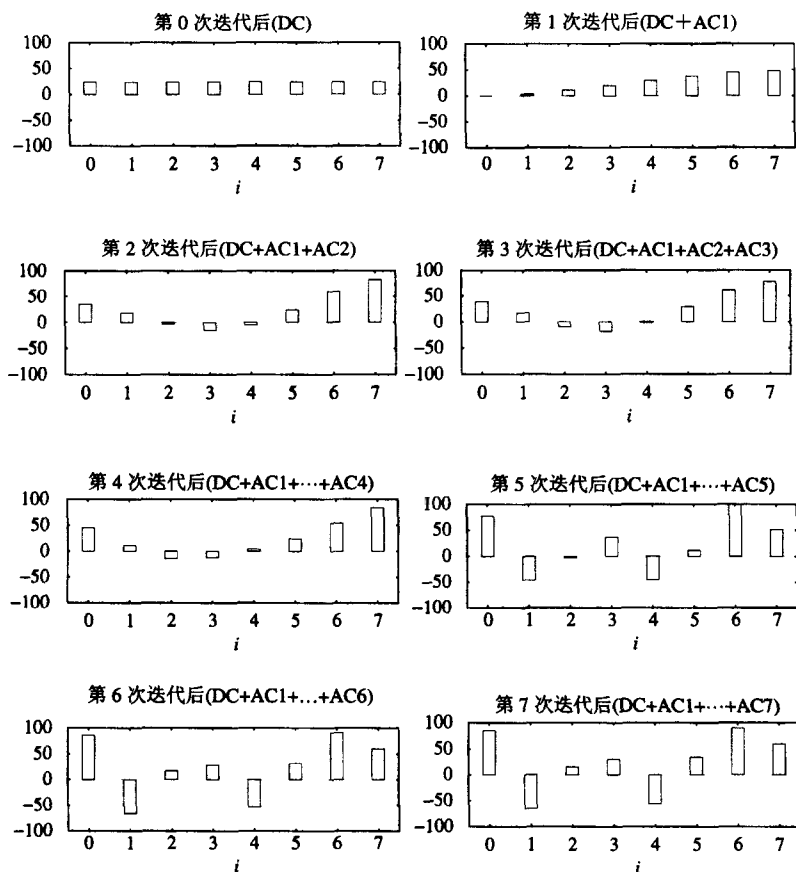


图 8-8 一维 IDCT 的一个例子

$$\text{迭代 0: } \tilde{f}(i) = \frac{C(0)}{2} \cdot \cos 0 \cdot F(0) = \frac{\sqrt{2}}{2.2} \cdot 1 \cdot 69 \approx 24.3$$

$$\begin{aligned} \text{迭代 1: } \tilde{f}(i) &= \frac{C(0)}{2} \cdot \cos 0 \cdot F(0) + \frac{C(1)}{2} \cdot \cos \frac{(2i+1)\pi}{16} \cdot F(1) \\ &\approx 24.3 + \frac{1}{2} \cdot (-49) \cdot \cos \frac{(2i+1)\pi}{16} \approx 24.3 - 24.5 \cdot \cos \frac{(2i+1)\pi}{16} \end{aligned}$$

$$\begin{aligned} \text{迭代 2: } \tilde{f}(i) &= \frac{C(0)}{2} \cdot \cos 0 \cdot F(0) + \frac{C(1)}{2} \cdot \cos \frac{(2i+1)\pi}{16} \cdot F(1) + \frac{C(2)}{2} \cdot \cos \frac{(2i+1)\pi}{8} \cdot F(2) \\ &\approx 24.3 - 24.5 \cdot \cos \frac{(2i+1)\pi}{16} + 37 \cdot \cos \frac{(2i+1)\pi}{8} \end{aligned}$$

经过初始迭代,  $\tilde{f}(i)$  为一个约等于 24.3 的常量, 这是  $f(i)$  中 DC 分量的复原。第一步迭代后,  $\tilde{f}(i) \approx 24.3 - 24.5 \cdot \cos \frac{(2i+1)\pi}{16}$ , 这是 DC 分量和第一个 AC 分量的和; 第二步迭代后,  $\tilde{f}(i)$  就是 DC 和 AC1、AC2 的和, 依此类推。如图 8-8 所示, IDCT 的求乘积之和过程最终重现 (重构) 了函数  $f(i)$ , 结果近似为

$$\tilde{f}(i)(i=0..7): 85 \quad -65 \quad 15 \quad 30 \quad -56 \quad 35 \quad 90 \quad 60$$

由上面的过程可知, 由整数的初值得到了整数的结果虽然通过中间运算的浮点数, 信号还是被精确地还原; 尽管情况不会总是这样, 不过最终结果却总是闭合的。

#### 4. 余弦基函数

要得到更好的分解, 其基函数应当是正交的 (orthogonal), 这样其中的冗余信息最少。

如果两个函数  $B_p(i)$  和  $B_q(i)$  满足下式, 它们就是正交的。

$$\sum_i [B_p(i) \cdot B_q(i)] = 0, p \neq q \quad (8.22)$$

两个函数  $B_p(i)$  和  $B_q(i)$  正交并且满足下式, 它们就是标准正交 (orthonormal) 的。

$$\sum_i [B_p(i) \cdot B_q(i)] = 1, p = q \quad (8.23)$$

标准正交性是我们希望得到的。有了这个特性, 信号在变换过程中就不会被放大。如果变换和其逆变换 (有时也称为正向变换 (forward transform) 和反向变换 (backward transform)) 都采用相同的基函数, 那么得到的信号将与原信号完全 (大致) 相同。

可以得出

$$\begin{aligned} \sum_{i=0}^7 \left[ \cos \frac{(2i+1) \cdot p\pi}{16} \cdot \cos \frac{(2i+1) \cdot q\pi}{16} \right] &= 0, p \neq q \\ \sum_{i=0}^7 \left[ \frac{C(p)}{2} \cos \frac{(2i+1) \cdot p\pi}{16} \cdot \frac{C(q)}{2} \cos \frac{(2i+1) \cdot q\pi}{16} \right] &= 1, p = q \end{aligned}$$

由此可见, DCT 中的余弦基函数确实是正交的, 取合适的常数  $C(p)$  和  $C(q)$  还可以使它们成为标准正交函数。(现在就可以理解 DCT 和 IDCT 的定义中的常数  $C(u)$  和  $C(v)$  为何看起来像是随意取的值了。)

回想前面图 8-7b 中的  $f_2(i)$ , 由于具有正交性, 只有  $F_2(2)$  (对于  $u=2$ ) 为非零值, 而其他 DCT 系数都为零。对某些信号在频域上的处理和分析, 这一点是很必要的, 因为这样就可以精确

地辨别出原信号的频域分量。

余弦基函数与三维笛卡儿空间中的基向量  $\vec{x}$ 、 $\vec{y}$ 、 $\vec{z}$ （或称三维向量空间）类似，这些向量是标准正交的，因为

$$\vec{x} \cdot \vec{y} = (1, 0, 0) \cdot (0, 1, 0) = 0$$

$$\vec{x} \cdot \vec{z} = (1, 0, 0) \cdot (0, 0, 1) = 0$$

$$\vec{y} \cdot \vec{z} = (0, 1, 0) \cdot (0, 0, 1) = 0$$

$$\vec{x} \cdot \vec{x} = (1, 0, 0) \cdot (1, 0, 0) = 1$$

$$\vec{y} \cdot \vec{y} = (0, 1, 0) \cdot (0, 1, 0) = 1$$

$$\vec{z} \cdot \vec{z} = (0, 0, 1) \cdot (0, 0, 1) = 1$$

216

任意点  $P = (x_p, y_p, z_p)$  都可以用一个向量  $\vec{OP} = (x_p, y_p, z_p)$  来表示，其中  $O$  为原点，该向量可被分解为  $x_p \cdot \vec{x} + y_p \cdot \vec{y} + z_p \cdot \vec{z}$ 。

如果把式 (8.19) 中求乘积之和的运算看作是离散余弦基函数（对确定的  $u$ ）与信号  $f(i)$  的点乘，那么 DCT 与笛卡儿投影之间的相似性就更加明显了。也就是说，要得到点  $P$  的  $x$  坐标，只需将  $P$  投影到  $x$  轴上。从数学上说，这相当于一次点乘运算  $\vec{x} \cdot \vec{OP} = x_p$ ，用同样的方法也可以得到  $y_p$  和  $z_p$ 。

现在，与图 8-7b 中的例子进行比较，对于笛卡儿空间中的点  $P = (0, 5, 0)$ ，只有  $y$  轴上的投影  $y_p = 5$ ，在  $x$  轴和  $z$  轴上的投影都是 0。

### 5. 二维基函数

对二维 DCT 函数来说，使用显示为  $8 \times 8$  图像的基。如图 8-9 所示，图中白色表示正值，黑色表示负值。为了获得 DCT 系数，只需求 64 个基图像中每一个与原图像对应的  $8 \times 8$  区域块的内积即可。这里我们考虑的是空间上的信号，而并非时间上的信号。对每个  $8 \times 8$  图像块进行如上操作，得到的 64 个乘积就是一个  $8 \times 8$  的空间频谱图像  $F(u, v)$ 。

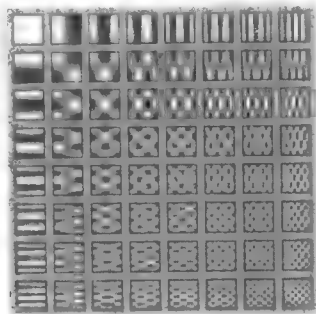


图 8-9 二维 DCT 基的图示

### 6. 二维分离基

考虑到计算速度，大多数软件的应用中都使用固定点的四则运算来计算 DCT 变换。正如能够数学上推导出快速傅里叶变换一样，也有快速 DCT。一些快速计算中将乘法转化为移位和加法后得到近似的系数，此外，还有一种更加简单的方法用来生成二维 DCT 的系数，这种方法将原问题通过因式分解转化为两个一维 DCT 变换。

217

当块的大小为 8 时，二维 DCT 可以被分解成顺序的两步一维 DCT。首先，对每一列进行一维 DCT，计算出一个中间函数  $G(i, v)$ ，这样就将列的维度转变到了频域，不过行维度还没有转变：



$$G(i, v) = \frac{1}{2} C(v) \sum_{j=0}^7 \cos \frac{(2j+1)v\pi}{16} f(i, j) \quad (8.24)$$

然后再次计算一维 DCT，这次将行维度替换成其对应的频域部分：

$$F(u, v) = \frac{1}{2} C(u) \sum_{i=0}^7 \cos \frac{(2i+1)u\pi}{16} G(i, v) \quad (8.25)$$

因为二维 DCT 的基函数是可分离的（ $i$  和  $j$  各自的独立函数相乘），所以这样的做法是可行的。可以看到，这样一个简单的转化节约了很多计算步骤，所需的迭代次数从  $8 \times 8$  变为了  $8+8$ 。

### 7. DCT 与 DFT 的比较

离散余弦变换[4]与离散傅里叶变换（DFT）是密切相关的，在信号处理领域，后者可能更常用一些。由于 DCT 相对比较简单并且在多媒体方面的应用更加广泛，因此本书主要介绍了 DCT。然而，我们不应当完全忽略 DFT。

一个连续信号的连续傅里叶变换  $\mathcal{F}$  定义如下：

$$\mathcal{F}(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \quad (8.26)$$

应用欧拉公式，可以得到

$$e^{ix} = \cos(x) + i \sin(x) \quad (8.27)$$

可见，连续傅里叶变换由无限个正弦项和余弦项的和组成。而计算机的计算是离散的，要将输入信号离散化，为此定义操作输入信号  $\{f_0, f_1, \dots, f_7\}$  的 8 个样本的 DFT 为

$$F_{\omega} = \sum_{x=0}^7 f_x \cdot e^{-\frac{2\pi i \omega x}{8}} \quad (8.28)$$

将正弦项和余弦项分开写，可得

$$F_{\omega} = \sum_{x=0}^7 f_x \cos\left(\frac{2\pi \omega x}{8}\right) - i \sum_{x=0}^7 f_x \sin\left(\frac{2\pi \omega x}{8}\right) \quad (8.29)$$

即使不给出 DCT 的明确定义，我们也可以猜到 DCT 可能是只包括 DFT 实数部分的变换。这样构造 DCT，使得 DCT 可以只包括 DFT 中的余弦基函数，因为只需产生一个与原输入信号对称的信号就可以将其虚部抵消掉。

因为正弦函数是奇函数，所以当信号被对称地扩展后，正弦项对结果的影响就会互相抵消。因此，8 个输入样本的 DCT 相当于 16 个样本的 DFT 和其对称样本的叠加，如图 8-10 所示。

218

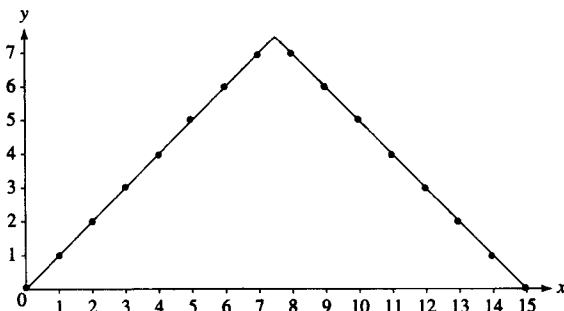


图 8-10 斜坡函数的对称扩展

经过对称扩展后, DCT 处理的是三角波, 而 DFT 则处理重复的斜坡函数。由于 DFT 不得不处理斜坡函数之间的连接造成的突变, 这就需要引入大量的高频分量。(更多关于 DCT 和 DFT 的讨论与比较请参考[4].)

表 8-1 给出的是计算得出的 DCT 系数和 DFT 系数。从中可以看出, 比起 DFT, DCT 的前几个系数包含了更多的信号信息。如果只用 DCT 和 DFT 的前三项来分别作为原斜坡函数的近似, 可以发现用 DCT 时的近似更加接近原函数, 两者的比较见图 8-11。

表 8-1 斜坡函数的 DCT 系数和 DFT 系数

斜坡函数	DCT	DFT
0	9.90	28.00
1	-6.44	-4.00
2	0.00	9.66
3	-0.67	-4.00
4	0.00	4.00
5	-0.20	-4.00
6	0.00	1.66
7	-0.51	-4.00

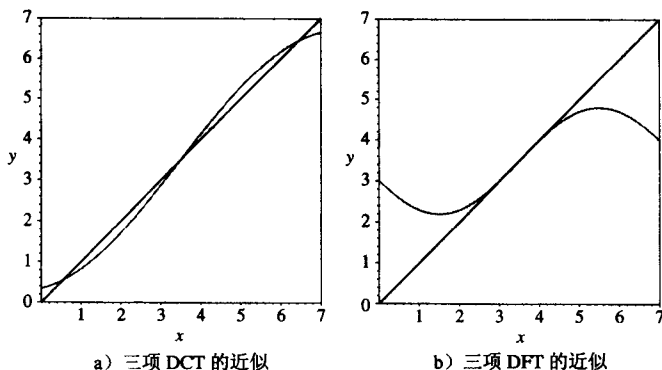


图 8-11 斜坡函数的近似

### \*8.5.2 Karhunen-Loève 变换

Karhunen-Loève 变换 (KLT) 是一种可逆的线性变换, 它应用了向量表述的统计学性质。Karhunen-Loève 变换主要的特性就是能够很好地解除输入的相关性, 为此, 它在 (减去平均值的) 数据附近找出一个  $n$  维椭球体, 该椭球体的长轴方向就是数据变化较大的方向。

想像一支被踩扁的雪茄的样子。描述雪茄的数据是雪茄各点在三维空间中的坐标值, 其长轴由一个统计程序找出来, 作为 KLT 的第一条轴。第二条重要的轴是通过被压扁的雪茄的水平轴, 它垂直于第一条轴。第三条轴垂直于前两条轴, 是竖直的。KLT 进行的就是这样的分析。

为了理解 KLT 的最优性, 考察输入向量  $X$  的自相关矩阵  $R_X$ , 其定义如下:

$$R_X = E[XX^T] \quad (8.30)$$

$$= \begin{bmatrix} R_X(1,1) & R_X(1,2) & \cdots & R_X(1,k) \\ R_X(2,1) & R_X(2,2) & \cdots & R_X(2,k-1) \\ \vdots & \vdots & \ddots & \vdots \\ R_X(k,1) & R_X(k-1,1) & \cdots & R_X(1,1) \end{bmatrix} \quad (8.31)$$

其中  $R_X(t,s) = E[X_t X_s]$  是自相关函数。我们的目的是找到一个变换  $T$ , 使得输出  $Y$  的各个分量都不相关。也就是说, 当  $t \neq s$  时,  $E[Y_t Y_s] = 0$ 。因此,  $Y$  的自相关矩阵的形式是正对角阵。

任何自相关矩阵都是对称和非负定义的矩阵, 因此它有  $k$  个正交的特征向量  $u_1, u_2, \dots, u_k$  和  $k$  个对应的非负实特征值  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k \geq 0$ 。Karhunen-Loève 变换的定义如下:

$$T = [u_1, u_2, \dots, u_k]^T \quad (8.32)$$

这样,  $Y$  的自相关矩阵就变为:

$$R_Y = E[YY^T] \quad (8.33)$$

$$= E[TXX^T T] \quad (8.34)$$

$$= TR_X T^T \quad (8.35)$$

$$= \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ 0 & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \lambda_k \end{bmatrix} \quad (8.36)$$

显然, 上面  $Y$  的自相关矩阵就是所期望的矩阵。因为它能够完全解除输入的相关性, 因此 KLT 是最优的。另外, 由于 KLT 依赖于输入向量自相关矩阵的计算, 因而它是数据相关的, 对每个数据集都要重新进行计算。

#### 例 8-5

为了说明 KLT 的运行机制, 以四个三维输入向量为例, 它们是  $x_1 = (4, 4, 5)$ ,  $x_2 = (3, 2, 5)$ ,  $x_3 = (5, 7, 6)$ ,  $x_4 = (6, 7, 7)$ 。为了找到所需的变换, 首先必须计算出输入的自相关矩阵。四个输入向量的均值为:

$$\mathbf{m}_x = \frac{1}{4} \begin{bmatrix} 18 \\ 20 \\ 23 \end{bmatrix}$$

使用下面的公式可以计算出自相关矩阵:

$$\mathbf{R}_X = \frac{1}{M} \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T - \mathbf{m}_x \mathbf{m}_x^T \quad (8.37)$$

上式中  $n$  是输入向量的数目。从这个公式可以得到:

$$\mathbf{R}_X = \begin{bmatrix} 1.25 & 2.25 & 0.88 \\ 2.25 & 4.50 & 1.50 \\ 0.88 & 1.50 & 0.69 \end{bmatrix}$$

221

$\mathbf{R}_X$  的特征值为  $\lambda_1 = 6.1963$ ,  $\lambda_2 = 0.2147$ ,  $\lambda_3 = 0.0264$ 。显然, 其中第一个分量是最重要的。特征值对应的特征向量为:

$$\mathbf{u}_1 = \begin{bmatrix} 0.4385 \\ 0.8471 \\ 0.3003 \end{bmatrix} \quad \mathbf{u}_2 = \begin{bmatrix} 0.4460 \\ -0.4952 \\ 0.7456 \end{bmatrix} \quad \mathbf{u}_3 = \begin{bmatrix} -0.7803 \\ 0.1929 \\ 0.5949 \end{bmatrix}$$

因此, KLT 可以由下面的矩阵得到:

$$\mathbf{T} = \begin{bmatrix} 0.4385 & 0.8471 & 0.3003 \\ 0.4460 & -0.4952 & 0.7456 \\ -0.7803 & 0.1929 & 0.5949 \end{bmatrix}$$

各个输入向量减去平均值向量后进行 KLT 变换, 得到结果如下:

$$\mathbf{y}_1 = \begin{bmatrix} -1.2916 \\ -0.2870 \\ -0.2490 \end{bmatrix} \quad \mathbf{y}_2 = \begin{bmatrix} -3.4242 \\ 0.2573 \\ 0.1453 \end{bmatrix}$$

$$\mathbf{y}_3 = \begin{bmatrix} 1.9885 \\ -0.5809 \\ 0.1445 \end{bmatrix} \quad \mathbf{y}_4 = \begin{bmatrix} 2.7273 \\ 0.6107 \\ -0.0408 \end{bmatrix}$$

由于  $T$  的各行是正交向量, 其逆变换就是其转置:  $T^{-1} = T^T$ 。通过逆关系, 由变换系数可以得出原向量。

$$x = T^T y + m_x \quad (8.38)$$

至于变换系数  $y_i$ , 前面几个分量的幅值通常都比其余的分量大得多。一般来说, 经过 KLT 变换后, 变换系数的大多数“能量”都集中到了前面几个分量里, 这叫做 KLT 的能量紧凑特性。

对于一个有  $n$  个分量的输入向量  $x$ , 如果对其输出向量  $y$  进行粗略的量化, 设置其后  $k$  个分量的值为零, 这个向量称作结果向量  $\hat{y}$ , KLT 变换能够使得原向量与其重现值之间均方差最小。

## 8.6 小波编码

### 8.6.1 简介

将输入信号分解成若干分量后, 我们便能够针对每个分量采用合适的编码方法, 以提高压缩性能。仍以一个和时间相关的信号  $f(t)$  为例 (以连续函数作为基础进行讨论是最好的)。传统的信号分解方法是傅里叶变换, 前面讨论的 DCT 是一种特殊的基于余弦的变换。如果同时基于正弦和余弦进行分析, 可以将结果用一个符号  $\mathcal{F}(\omega)$  表示, 这个函数由式 (8.26) 得到, 它的值是复数, 频率  $\omega$  是实数。进行这样的分解后, 在频域上有很高的分辨率。然而由于正弦曲线理论上在时间上是无限的, 这样的分解不能得到瞬时 (temporal) 结果。

另一种近年来常用的分解方法是小波变换 (wavelet transform)。它采用一组称为小波的基函数来表示信号, 可以在时域和频域上都得到很好的分辨率。

小波变换有两种类型: 连续小波变换 (Continuous Wavelet Transform, CWT) 和离散小波变换 (Discrete Wavelet Transform, DWT)。CWT 应用于在实数域上平方可积的函数  $f(x)$ , 也就是  $\int [f(x)]^2 dx < \infty$ , 在数学上也可以写作  $f(x) \in L^2(R)$ 。

另一种小波变换 DWT 用于处理输入信号的离散采样。DWT 和其他的离散线性变换。(如 DFT 和 DCT) 类似, 在图像处理和压缩中非常有用。

在讨论小波理论之前, 我们先介绍一个最简单的小波变换的例子, 以便让读者对小波变换有一个直观的印象。这种小波变换叫作 Haar 小波变换, 它能得出一个实数序列的平均值和差值。

序列的多分辨率分析是指重复地计算出每一步的序列平均值和差值, 并将结果记录下来。对于图像而言, 这就相当于产生越来越小的简化图像, 每一步的图像都只有上一步的 1/4 大小, 同时还记录下其与平均值的差值。将全尺寸图像、1/4 图像、1/16 图像等排列起来, 会形成一个金字塔, 这些图像的集合与图像之间的差值就构成了多分辨率分析。

#### 例 8-6 一个小波变换

小波变换的目的是为了更好地进行压缩, 将输入信号分解为易于处理、有特殊含义或者可被忽略的分量。当然, 变换后的分量必须能够近似地重现原信号。假设有如下输入序列:

$$\{x_{n,i}\} = \{10, 13, 25, 26, 29, 21, 7, 15\} \quad (8.39)$$

其中,  $i \in [0, \dots, 7]$  为像素编号,  $n$  代表当前的金字塔层级。对于这个序列来说, 在顶层  $n=3$ , 对  $n=2$ , 1 和 0, 还需再构造三个数列。在每个层级上, 变换后的信号数列的前几个元素中所保留的信息较少。在金字塔的层级  $n=0$  时, 其第一个元素是数列的平均值, 其他详细信息则存储在剩余的元素中。

定义如下的变换, 它将原数列替换为其相邻两个元素的平均值  $x_{n-1,i}$  和两两差值  $d_{n-1,i}$ :

$$x_{n-1,i} = \frac{x_{n,2i} + x_{n,2i+1}}{2} \quad (8.40)$$

$$d_{n-1,i} = \frac{x_{n,2i} - x_{n,2i+1}}{2} \quad (8.41) \quad \boxed{223}$$

由于所取的平均和差值都取自全部的序列中下标为偶数的连续两个元素，所以集合  $\{x_{n-1,i}\}$  和  $\{d_{n-1,i}\}$  中元素的数目都正好是原数列中元素数目的一半。将数列  $\{x_{n-1,i}\}$  和  $\{d_{n-1,i}\}$  连接起来就可以得到一个与原数列长度相同的新数列，结果如下：

$$\{x_{n-1,i}, d_{n-1,i}\} = \{11.5, 25.5, 25, 11, -1.5, -0.5, 4, -4\} \quad (8.42)$$

现在的层级是  $n-1=2$ ，该数列与输入数列有相同数目的元素，变换并没有增加数据量。因为上面数列的前一半是原数列的平均值，所以可以把它看成是原信号的粗略近似。

该数列的后一半则可以看作是前一半的细化或近似误差。这一细化数列中大多数的数值都比原数列的小得多，这样，大部分的能量就被有效地集中到了前面一半数列中。因此，可以用更少的位数来存储  $\{d_{n-1,i}\}$ 。

可以很容易地证明，用下面的关系式可以从变换后的数列重现原数列。

$$\begin{aligned} x_{n,2i} &= x_{n-1,i} + d_{n-1,i} \\ x_{n,2i+1} &= x_{n-1,i} - d_{n-1,i} \end{aligned} \quad (8.43)$$

上面的变换就是离散 Haar 小波变换。应用缩放函数和小波函数可以完成求平均和取差的操作，图 8-12 所示为 Haar 小波变换中的这两个函数。

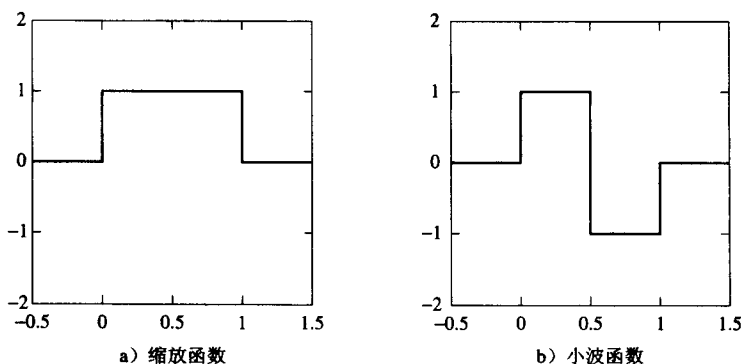


图 8-12 Haar 小波变换

224

将同样的变换用于  $\{x_{n-1,i}\}$ ，可以得到下一层级的近似值  $x_{n-2,i}$  和细化值  $d_{n-2,i}$ ：

$$\{x_{n-2,i}, d_{n-2,i}, d_{n-1,i}\} = \{18.5, 18, -7, 7, -1.5, -0.5, 4, -4\} \quad (8.44)$$

以上就是多分辨率分析的基本思想。现在就可以在三个不同的尺度上来研究输入信号，按照所需的细节而从一个尺度转到另一个。分析过程可以重复  $n$  次，直到在近似值数列中只剩下一个元素。在本例中， $n=3$ ，最终得到的数列为

$$\{x_{n-3,i}, d_{n-3,i}, d_{n-2,i}, d_{n-1,i}\} = \{18.25, 0.25, -7, 7, -1.5, -0.5, 4, -4\} \quad (8.45)$$

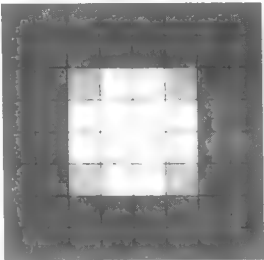
现在才知道例子中  $n$  之所以是 3，正是因为得到最终的结果需要经过三次变换。

数值 18.25 是原信号的最粗略的近似，也正是原数列中所有元素的平均值。从本例中也可看出，变换的计算的复杂度与输入数列的元素数目  $N$  成正比，即  $O(N)$ 。

将一维 Haar 小波变换扩展到二维很简单，只需对二维输入的行和列分别进行一维变换即可。下面将描述如何在图 8-13 所示的 8×8 输入图像上应用二维 Haar 变换。

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	63	127	127	63	0	0
0	0	127	255	255	127	0	0
0	0	127	255	255	127	0	0
0	0	63	127	127	63	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

a) 像素值



b) 一张 8×8 的图像

图 8-13 二维 Haar 小波变换的输入图像

例 8-7 二维 Haar 变换

225

这个二维 Haar 变换的例子不仅用来说明如何在二维输入上应用小波变换，同时也指出了变换系数的意义。不过，举这个例子的目的只是使读者对进行一般的二维小波变换时的操作有一个直观认识。后续小节将具体介绍正向和逆向二维小波变换的算法，并且会给出一个更加详细的采用更加复杂的小波的例子。

二维 Haar 小波变换

开始，我们先对输入的每一行应用一维 Haar 小波变换，前两行和后两行都为零，对剩余的几行进行取平均和求差值运算后，得到如图 8-14 所示的中间输出结果。

接下来对中间结果的各列进行相同的一维 Haar 变换，这就完成了一个层级的二维 Haar 变换。图 8-15 给出的是得到的系数。

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	95	95	0	0	-32	32	0
0	191	191	0	0	-64	64	0
0	191	191	0	0	-64	64	0
0	95	95	0	0	-32	32	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

图 8-14 二维 Haar 小波变换的中间输出结果

0	0	0	0	0	0	0	0
0	143	143	0	0	-48	48	0
0	143	143	0	0	-48	48	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	-48	-48	0	0	16	-16	0
0	48	48	0	0	-16	16	0
0	0	0	0	0	0	0	0

图 8-15 第一层级二维 Haar 小波变换的输出结果

226

将所得结果划分为四个象限，左上象限包含的是水平方向和竖直方向上的平均系数，因此它可以被看作对原图像的低通滤波，滤掉了高频的尖锐信号，而保留了低空间频率的平滑信息。

右上象限包含的是水平方向上的差值在竖直方向上的平均值，可以表示原图像的垂直边缘信息。同样，左下象限包含的是水平方向上的平均值在竖直方向上的差值，可以表示原图像的水平边缘。右下象限包含的是水平方向和竖直方向上的差值，这个象限的系数表示的是对角边缘。

图 8-16 以图像的形式更加清楚地表示了上面的阐述，其中亮的像素编码为正值，暗的像素编码为负值。

二维 Haar 变换的逆变换首先使用式 (8.43) 转化出列，然后再转化出行。

## \*8.6.2 连续小波变换

使用小波变换的初衷是,利用一系列基函数把时域上的信号分解成频域和时域上变量的函数。傅里叶变换的主要作用是控制空域变化较大的信号,对时域上变化较大的信号,傅里叶变换作用不大。而小波变换的目的是处理图像不同部分的内容。

例如,图像的一部分可能纹理显著,这是图像的高频部分;而另一部分可能很光滑,这是图像的低频部分。对这种情况,人们自然会想到把图像分成不同部分,之后连续使用傅里叶变换。把不同时段频率不同的函数分开处理,称为短时(或 Windowed)傅里叶变换。然而,新近出现的小波变换相对而言更加灵活。

为了深入了解连续小波变换,我们需要考虑物理学上 Heisenberg 不确定原则。对信号处理来说,在减少函数的频率和时间宽度的准确性之间有一个权衡。一般而言,我们很难用有效的基函数使得二者同时精确。例如,正弦波在频域上非常精确但是在宽度上却是无限的。

以下是一个快速衰减并且在频域上有限的高斯函数。

$$f(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-t^2}{2\sigma^2}} \quad (8.46)$$

参数  $\sigma$  是高斯函数的幅值。

从该函数可以派生出另一个函数,称为  $\psi(t)$ ,如图 8-17a 所示,它的波形类似于墨西哥帽。 $\psi(t)$  显然在时间上是有限的。它的方程如下所示:

$$\psi(t) = \frac{1}{\sigma^3\sqrt{2\pi}} \left[ e^{\frac{-t^2}{2\sigma^2}} \left( \frac{t^2}{\sigma^2} - 1 \right) \right] \quad (8.47)$$

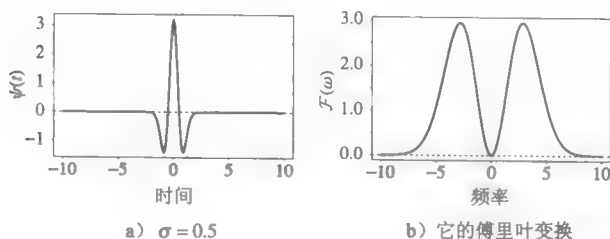


图 8-17 墨西哥帽形小波

我们对  $\psi(t)$  做傅里叶变换可以得到它的频率特性。这可以用下式表示:

$$\mathcal{F}(\omega) = \omega^2 e^{-\frac{\sigma^2 \omega^2}{2}} \quad (8.48)$$

图 8-17b 显示了如下函数,其候选小波式 (8.47) 在频域内确实是有限的。

一般来说,小波是具有零值的函数  $\psi \in L^2(R)$ 。

$$\int_{-\infty}^{+\infty} \psi(t) dt = 0 \quad (8.49)$$

上式满足某些条件,使得它能用于多分辨率分解。这些条件可以保证在分解之后,我们能够对图

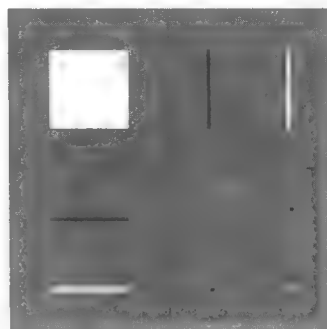


图 8-16 小波变换的一个简单的图形示意

228 像的局部进行缩放, 就像能够对地图进行缩放一样。

式(8.49)称为小波的成立条件(admissibility condition)。一个总值为0的函数, 必然在0轴上下波动。同样, 从式(8.26)可以看出, 在 $\omega=0$ 时对 $\psi(t)$ 进行傅里叶变换后, DC分量为0。也可以说,  $\psi(t)$ 0时刻 $M_0$ 的值为0, 第 $p$ 个时刻的值为:

$$M_p = \int_{-\infty}^{\infty} t^p \psi(t) dt \quad (8.50)$$

$\psi$ 函数被规范为 $\|\psi\|=1$ , 而且集中在 $t=0$ 附近。把下述母函数(mother wavelet)放大、变换之后, 我们可以得到一系列小波函数。如下所示:

$$\psi_{s,u}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{t-u}{s}\right) \quad (8.51)$$

只要 $\psi(t)$ 函数是规格化的, 那么 $\psi_{s,u}(t)$ 也是规格化的。

连续小波变换(CWT)  $f \in L^2(R)$ 在时间 $u$ 、缩放空间 $s$ 上的定义如下:

$$\mathcal{W}(f, s, u) = \int_{-\infty}^{+\infty} f(t) \psi_{s,u}(t) dt \quad (8.52)$$

对一维信号的小波变换结果是关于缩放空间 $s$ 和偏移量 $u$ 的二维函数。

重要的是, 和式(8.26)不同, 傅里叶变换得出的是正弦曲线, 而式(8.52)却说明小波变换却不一定得出正弦曲线。小波变换确定了一系列变换函数必须遵守的规则, 然后可以创建各种符合规则的函数, 用来满足不同的应用。

DCT变换的结果可以看作是一系列基函数的乘积, 那么这里小波变换 $\mathcal{W}$ 的结果就是从母函数 $\psi(t)$ 用缩放和平移得出的基函数相互作用的和。

由于母函数 $\psi(t)$ 必须是一个振荡函数, 所以它是一个波。那么为什么是小波呢? 式(8.52)中的 $s$ 是空域上的分析参数。我们可以取某些 $s$ 的值来看信号在这些值上的分量如何。为了使函数快速衰减, 除了选择 $s$ 的值以外, 我们必须选择一个衰减性能和 $s$ 的指数相当的母函数 $\psi(t)$ 。

从式(8.52)可以看出, 如果 $\psi(t)$ 中接近 $n$ 的时刻值都为0(或者很小, 近似0), 那么CWT的系数 $\mathcal{W}(f, s, u)$ 在 $u=0$ 附近呈 $s^{n+2}$ 泰勒级数(见练习9)。这是我们期望的好的函数中的频率局域化。

对信号的不同部分进行不同比例的小波变换之后, 我们可以得到小波系数。令人激动的是, 如果我们压缩小波, 使它小到能包括一部分多项式最高次数为 $n$ 的多项式 $f(t)$ , 此小波以及更小的小波的系数将为0。小波必须有某种顺序的趋零瞬间的条件, 是母小波具有某种数学上规律性约束的特征。

连续小波变换的逆变换如下:

229

$$f(t) = \frac{1}{C_\psi} \int_0^{+\infty} \int_{-\infty}^{+\infty} \mathcal{W}(f, s, u) \frac{1}{\sqrt{s}} \psi\left(\frac{t-u}{s}\right) \frac{1}{s^2} du ds \quad (8.53)$$

其中

$$C_\psi = \int_0^{+\infty} \frac{|\Psi(\omega)|^2}{\omega} d\omega < +\infty \quad (8.54)$$

$\Psi(\omega)$ 是 $\psi(t)$ 的傅里叶变换。式(8.54)是其成立条件的另一种表示。

但CWT的问题是(8.52)性能并不好: 大部分的小波都是从简单地数值计算中得出的, 而不是解析的。这样的结果是一系列无穷的缩放和平移函数, 而通常对采样函数的分析都不需要这样的操作(如在图像处理中)。因此, 我们把CWT转换到离散数域中。



### \*8.6.3 离散小波变换

离散小波也是从一个母函数中派生而来，但是平移和缩放都是离散的。

#### 1. 多分辨率分析和离散小波变换

连续时域上的小波和离散时域上的滤波器组 (filter banks) 之间的联系是多分辨率分析。我们就在这一框架内讨论 DWT。Mallat[5]中提到，可以从母函数通过伸缩和平移得到一系列  $L^2(R)$  的正交基。如下，其中  $Z$  表示一组整数。

$$\left\{ \psi_{j,n}(t) = \frac{1}{\sqrt{2^j}} \psi \left( \frac{t - 2^j n}{2^j} \right) \right\}_{(j,n) \in Z^2} \quad (8.55)$$

这叫做二进缩放和平移，效果类似对地图以因子  $Z$  进行缩小。(我们看余弦函数的图形，从时间  $0 \sim 2\pi$  的  $\cos(t)$  和  $\cos(t/2)$ ，当  $\cos(t)$  过了一个周期时， $\cos(t/2)$  只有半个周期。函数  $\cos(2^{-1}t)$  是一个展宽函数，因而有一个更大的尺度。)

注意，缩放比例的变化和总比例  $2^j$  总是保持一致，以此保证低分辨率的图形也有恰当的比例。通常来说，较大的  $j$  值对应的图像质量较低。

多分辨率分析能够针对不同场景，自适应的呈现波形中的某些细节。Mallat[6]首先提出来的八阶分解 Octave Decomposition，把信号分解成近似和细节两部分。近似部分还可以根据不同精度需要继续分解成近似和细节部分。小波变换能够使得在分辨率为  $2^{-j}$  时的近似值能够计算出更粗糙的近似  $2^{-(j+1)}$ 。

小波能够用来表示细节部分的信息。平均信息主要由母函数的一种对偶确定，叫做缩放函数  $\phi(t)$ 。

小波理论的主要思想是：在分辨率为  $j$  的情况下，可以由一系列标号为  $n$  的平移组成一个基。有趣的是，组成  $j+1$  次变换的基函数能够用权重之和乘上第  $j$  层的基函数表示，缩放函数的系数必须是有限的。

230

$j+1$  级 (就是 3，或 1/8 级) 的小波基函数由缩放函数及其平移值得到，第  $j$  级 (就是 2，或 1/4 级) 的小波基函数则由缩放函数  $\phi$  的一组平移值、以及母函数  $\phi$  的平移组得到。缩放函数描述平滑或近似的信息，小波描述其丢掉的 (即细节) 信息。

对函数  $\phi$  的缩放平移可以表示成上一层变换函数的加权和，所以缩放函数必须满足所谓的“伸缩等式” [7]：

$$\phi(t) = \sum_{n \in Z} \sqrt{2} h_0[n] \phi(2t - n) \quad (8.56)$$

方括号里的表达式源于滤波器理论。缩放方程表明要找的函数必须是从母函数通过平移、缩放之后的不同变体函数的和。式 (8.56) 说明了缩放函数必须满足的条件，同时还给出了缩放矢量  $h_0$  的定义。

不仅是缩放函数本身，小波基函数也可以表示成缩放函数的和：

$$\psi(t) = \sum_{n \in Z} \sqrt{2} h_1[n] \phi(2t - n) \quad (8.57)$$

小波基函数的系数矢量  $h_1$ ，可以由缩放函数的矢量得出，并且小波基函数也可以从缩放函数得出：

$$\psi(t) = \sum_{n \in Z} (-1)^n h_0[1-n] \phi(2t - n) \quad (8.58)$$

就是说，小波基函数和缩放函数有相同的性质。事实上，式 (8.56) 也使用了同样的系数，只不过用了相反的顺序和符号。

考虑性能因素，实际应用中式 (8.56) 和式 (8.57) 中的项越少越好，因此我们希望矢量  $h_1$  和  $h_0$  中项越少越好。缩放函数的作用是对信号进行缩放、平滑。所以它的实际效果类似于低通滤波

器,滤去高频部分。滤波函数对一个在  $t=0$  时刻出现的脉冲的作用,可以通过矢量  $h_0[n]$  反映,因此它又叫做低通滤波冲激响应因子。一个完整的离散信号是由从 0 开始,幅度为离散值的脉冲组成。

因此,只需要离散低通滤波脉冲响应因子  $h_0[n]$  就能描述一个 DWT。 $h_0[n]$  描述近似部分,高通滤波脉冲响应因子  $h_1[n]$  描述信号细节部分。它能够从  $h_0[n]$  推导出:

$$h_1[n] = (-1)^n h_0[1-n] \quad (8.59)$$

冲激响应中系数的个数又称为滤波器的 tap 数。如果  $h_0[n]$  有有限个非无穷的项,那么得出的小波叫做紧致小波。通常,有些  $h_0[n]$  还可以具有正交和规范的性质。 $h_0[n]$  和  $h_1[n]$  又分别称为低通和高通滤波器。

为了重建原始信号,我们需要一个逆滤波器,也叫做合成滤波器。对于正交的小波来说,正变换和逆变换可以互换。分析滤波器和合成滤波器是相同的。

更弱的情况是小波不正交,分解和合成所用的小波双正交。此时分解小波和综合小波不相同,分别记为  $\tilde{h}_0[n]$  和  $\tilde{h}_1[n]$ 。我们同时需要  $\tilde{h}_0[n]$  和  $h_0[n]$  来描述一个双正交的小波。同样,高通滤波器可以用低通滤波器来表示:

$$h_1[n] = (-1)^n \tilde{h}_0[1-n] \quad (8.60)$$

$$\tilde{h}_1[n] = (-1)^n h_0[1-n] \quad (8.61)$$

表 8-2 和表 8-3 给出了常用的正交和双正交的小波。表中的“开始索引”表示式 (8.60) 和式 (8.61) 中  $n$  的起始数字。

表 8-2 正交小波滤波器

小 波	拍 数	开始索引	系 数
Haar	2	0	[0.707, 0.707]
Daubechies 4	4	0	[0.483, 0.837, 0.224, -0.129]
Daubechies 6	6	0	[0.332, 0.807, 0.460, -0.135, -0.085, 0.0352]
Daubechies 8	8	0	[0.230, 0.715, 0.631, -0.028, -0.187, 0.031, 0.033, -0.011]

图 8-18 是一个一维二元小波变换器的示意图。这里  $x[n]$  表示要处理的离散信号。 $\downarrow 2$  表示每隔一秒的采样,而  $\uparrow 2$  表示恢复出采样间隔。重构变换得出  $y[n]$ 。

表 8-3 双正交小波滤波器

小 波	滤波器	拍数	开始索引	系 数
Antonini 9/7	$h_0[n]$	9	-4	[0.038, -0.024, -0.111, 0.377, 0.853, 0.377, -0.111, -0.024, 0.038]
	$\tilde{h}_0[n]$	7	-3	[-0.065, -0.041, 0.418, 0.788, 0.418, -0.041, -0.065]
Villa 10/18	$h_0[n]$	10	-4	[0.029, 0.0000824, -0.158, 0.077, 0.759, 0.759, 0.077, -0.158, 0.0000824, 0.029]
	$\tilde{h}_0[n]$	18	-8	[0.000954, -0.00000273, -0.009, -0.003, 0.031, -0.014, -0.086, 0.163, 0.623, 0.623, 0.163, -0.086, -0.014, 0.031, -0.003, -0.009, -0.00000273, 0.000954]
Brislaw	$h_0[n]$	10	-4	[0.027, -0.032, -0.241, 0.054, 0.900, 0.900, 0.054, -0.241, -0.032, 0.027]
	$\tilde{h}_0[n]$	10	-4	[0.020, 0.024, -0.023, 0.146, 0.541, 0.541, 0.146, -0.023, 0.024, 0.020]

在分解过程中,每一步  $x[n]$  被分解成同样长度的另一个序列,前面的部分是信号的近似,后面的部分是细节部分。对一个  $N$ -tap 的滤波器,在下面的序列中,

$$\{x[n]\} \rightarrow y[n] = \left\{ \sum_j x[j]h_0[n-j]; \sum_j x[j]h_1[n-j] \right\} \quad (8.62)$$

奇数部分被丢弃。式(8.62)中移位系数上的总和叫做卷积。

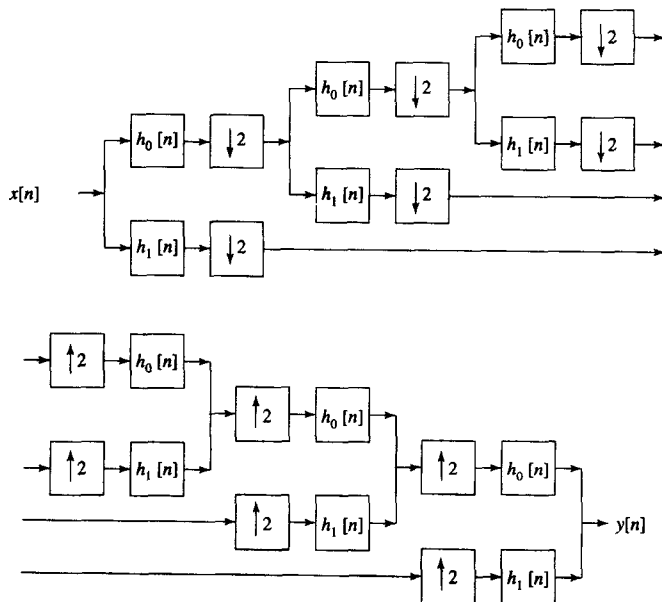


图 8-18 一维双通道小波变换示意图

## 2. 二维离散小波变换

把小波变换扩展到二维空间是很简单的。如果一个二维缩放函数能够分解成两个一维的缩放函数，则该二维缩放函数称为“可分解的”：

$$\phi(x, y) = \phi(x)\phi(y) \quad (8.63) \quad \boxed{233}$$

为了简化，本小节只考虑可分解的小波变换。而且本部分假设图像的长和宽都是 2 的幂。

考虑一个  $N \times N$  的图像，二维小波变换过程如下：

1) 对图像的每一行和  $h_0[n]$  和  $h_1[n]$  做卷积，丢弃结果中奇数部分，把剩下的两个数列组合成一个数列。

2) 对所有行做上述变换之后，把每一列和  $h_0[n]$  和  $h_1[n]$  做卷积，同样丢弃奇数部分的数据，把剩下的数列合成一个数列。

上述两个步骤是 DWT 的操作之一，现在，变换后的图像有四个部分组成：LL、HL、LH、HH，如图 8-19a 所示。在一维变换中，LL 子带还可以继续进行下一步分解。这个过程可以一直持续，直到足够多的分解，或者直到 LL 子带只有一个元素为止。二维的分解如图 8-19b 所示。

逆变换可以把分解的过程逆转：

- 1) 把变换后的图像中的每一列，分成低通和高通系数。在采样的间隔中插入 0。
- 2) 将低通部分和  $h_0[n]$  做卷积，高通部分和  $h_1[n]$  做卷积，求其和。
- 3) 所有的列都处理之后，将每一行都拆成低通和高通两部分，在采样间隔中插入 0。
- 4) 将低通部分和  $h_0[n]$  做卷积，高通部分和  $h_1[n]$  做卷积，求其和。

如果是用双正交滤波器，那么重现过程中  $h_0[n]$  和  $h_1[n]$  分别被  $\tilde{h}_0[n]$  和  $\tilde{h}_1[n]$  取代。

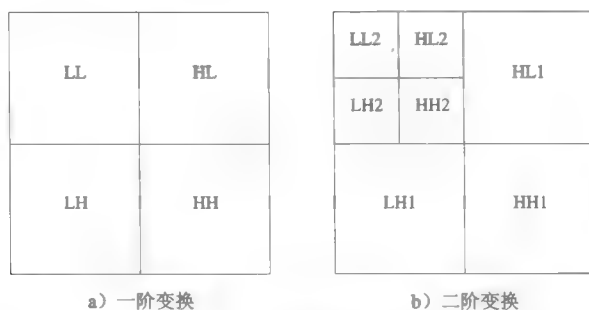


图 8-19 二维离散小波变换

**例 8-8**

输入图像是一个部分采样的 Lena 图像, 如图 8-20 所示, 大小为  $16 \times 16$ 。采用的是表 8-3 中给出的 Antonini 9/7 滤波器。

在开始之前, 我们要用式 (8.60) 和式 (8.61) 所示的方法, 计算分解和重现要用的高通和低通滤波器, 得到如下结果:

$$\begin{aligned} h_1[n] &= [-0.065, 0.041, 0.418, -0.788, 0.418, 0.041, -0.065] \\ \tilde{h}_1[n] &= [-0.038, -0.024, 0.111, 0.377, -0.853, 0.377, 0.111, -0.024, -0.038] \end{aligned} \quad (8.64)$$

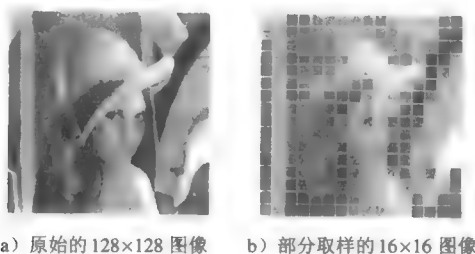


图 8-20 Lena

输入图像的数字表示如下:

$$I_{00}(x, y) = \begin{bmatrix} 158 & 170 & 97 & 104 & 123 & 130 & 133 & 125 & 132 & 127 & 112 & 158 & 159 & 144 & 116 & 91 \\ 164 & 153 & 91 & 99 & 124 & 152 & 131 & 160 & 189 & 116 & 106 & 145 & 140 & 143 & 227 & 53 \\ 116 & 149 & 90 & 101 & 118 & 118 & 131 & 152 & 202 & 211 & 84 & 154 & 127 & 146 & 58 & 58 \\ 95 & 145 & 88 & 105 & 188 & 123 & 117 & 182 & 185 & 204 & 203 & 154 & 153 & 229 & 46 & 147 \\ 101 & 156 & 89 & 100 & 165 & 113 & 148 & 170 & 163 & 186 & 144 & 194 & 208 & 39 & 113 & 159 \\ 103 & 153 & 94 & 103 & 203 & 136 & 146 & 92 & 66 & 192 & 188 & 103 & 178 & 47 & 167 & 159 \\ 102 & 146 & 106 & 99 & 99 & 121 & 39 & 60 & 164 & 175 & 198 & 46 & 56 & 56 & 156 & 156 \\ 99 & 146 & 95 & 97 & 144 & 61 & 103 & 107 & 108 & 111 & 192 & 62 & 65 & 128 & 153 & 154 \\ 99 & 140 & 103 & 109 & 103 & 124 & 54 & 81 & 172 & 137 & 178 & 54 & 43 & 159 & 149 & 174 \\ 84 & 133 & 107 & 84 & 149 & 43 & 158 & 95 & 151 & 120 & 183 & 46 & 30 & 147 & 142 & 201 \\ 58 & 153 & 110 & 41 & 94 & 213 & 71 & 73 & 140 & 103 & 138 & 83 & 152 & 143 & 128 & 207 \\ 56 & 141 & 108 & 58 & 92 & 51 & 55 & 61 & 88 & 166 & 58 & 103 & 146 & 150 & 116 & 211 \\ 89 & 115 & 188 & 47 & 113 & 104 & 56 & 67 & 128 & 155 & 187 & 71 & 153 & 134 & 203 & 95 \\ 35 & 99 & 151 & 67 & 35 & 88 & 88 & 128 & 140 & 142 & 176 & 213 & 144 & 128 & 214 & 100 \\ 89 & 98 & 97 & 51 & 49 & 101 & 47 & 90 & 136 & 136 & 157 & 205 & 106 & 43 & 54 & 76 \\ 44 & 105 & 69 & 69 & 68 & 53 & 110 & 127 & 134 & 146 & 159 & 184 & 109 & 121 & 72 & 113 \end{bmatrix}.$$

$I$  表示像素值,  $I$  的下标分别表示当前变换的级数和在本级变换中步数。我们首先将第一行分别和  $h_0[n]$ 、 $h_1[n]$  做卷积, 丢弃结果集中的奇数部分。操作的结果如下:

$$(I_{00}(:, 0) * h_0[n]) \downarrow 2 = [245, 156, 171, 183, 184, 173, 228, 160]$$

$$(I_{00}(:, 0) * h_1[n]) \downarrow 2 = [-30, 3, 0, 7, -5, -16, -3, 16]$$

其中冒号表示整行。读者可以自行用 MATLAB 验证这个结果。

下面,我们将以上两行合在一起。

$$[245, 156, 171, 183, 184, 173, 228, 160, -30, 3, 0, 7, -5, -16, -3, 16]$$

和一维 Haar 变换一样,现在大部分的能量都集中在结果的前半部分。继续计算剩下的行,得到如下结果:

$$I_{11}(x, y) = \begin{bmatrix} 245 & 156 & 171 & 183 & 184 & 173 & 228 & 160 & -30 & 3 & 0 & 7 & -5 & -16 & -3 & 16 \\ 239 & 141 & 181 & 197 & 242 & 158 & 202 & 229 & -17 & 5 & -20 & 3 & 26 & -27 & 27 & 141 \\ 195 & 147 & 163 & 177 & 288 & 173 & 209 & 106 & -34 & 2 & 2 & 19 & -50 & -35 & -38 & -1 \\ 180 & 139 & 226 & 177 & 274 & 267 & 247 & 163 & -45 & 29 & 24 & -29 & -2 & 30 & -101 & -78 \\ 191 & 145 & 197 & 198 & 247 & 230 & 239 & 143 & -49 & 22 & 36 & -11 & -26 & -14 & 101 & -54 \\ 192 & 145 & 237 & 184 & 135 & 253 & 169 & 192 & -47 & 38 & 36 & 4 & -58 & 66 & 94 & -4 \\ 176 & 159 & 156 & 77 & 204 & 232 & 51 & 196 & -31 & 9 & -48 & 30 & 11 & 58 & 29 & 4 \\ 179 & 148 & 162 & 129 & 146 & 213 & 92 & 217 & -39 & 18 & 50 & -10 & 33 & 51 & -23 & 8 \\ 169 & 159 & 163 & 97 & 204 & 202 & 85 & 234 & -29 & 1 & -42 & 23 & 37 & 41 & -56 & -5 \\ 155 & 153 & 149 & 159 & 176 & 204 & 65 & 236 & -32 & 32 & 85 & 39 & 38 & 44 & -54 & -31 \\ 145 & 148 & 158 & 148 & 164 & 157 & 188 & 215 & -55 & 59 & -110 & 28 & 26 & 48 & -1 & -64 \\ 134 & 152 & 102 & 70 & 153 & 126 & 199 & 207 & -47 & 38 & 13 & 10 & -76 & 3 & -7 & -76 \\ 127 & 203 & 130 & 94 & 171 & 218 & 171 & 228 & 12 & 88 & -27 & 15 & 1 & 76 & 24 & 85 \\ 70 & 188 & 63 & 144 & 191 & 257 & 215 & 232 & -5 & 24 & -28 & -9 & 19 & -46 & 36 & 91 \\ 129 & 124 & 87 & 96 & 177 & 236 & 162 & 77 & -2 & 20 & -48 & 1 & 17 & -56 & 30 & -24 \\ 103 & 115 & 85 & 142 & 188 & 234 & 184 & 132 & -37 & 0 & 27 & -4 & 5 & -35 & -22 & -33 \end{bmatrix}$$

下面对列进行变换,将第一列分别和  $h_0[n]$ 、 $h_1[n]$  做卷积,并丢弃奇数部分:

$$(I_{11}(0,:) * h_0[n]) \downarrow 2 = [353, 280, 269, 256, 240, 206, 160, 153]^T$$

$$(I_{11}(0,:) * h_1[n]) \downarrow 2 = [-12, 10, -7, -4, 2, -1, 43, 16]^T$$

继续处理剩下的列,得到如下:

$$I_{12}(x, y) = \begin{bmatrix} 353 & 212 & 251 & 272 & 281 & 234 & 308 & 289 & -33 & 6 & -15 & 5 & 24 & -29 & 38 & 120 \\ 280 & 203 & 254 & 250 & 402 & 269 & 297 & 207 & -45 & 11 & -2 & 9 & -31 & -26 & -74 & 23 \\ 269 & 202 & 312 & 280 & 316 & 353 & 337 & 227 & -70 & 43 & 56 & -23 & -41 & 21 & 82 & -81 \\ 256 & 217 & 247 & 155 & 236 & 328 & 114 & 283 & -52 & 27 & -14 & 23 & -2 & 90 & 49 & 12 \\ 240 & 221 & 226 & 172 & 264 & 294 & 113 & 330 & -41 & 14 & 31 & 23 & 57 & 60 & -78 & -3 \\ 206 & 204 & 201 & 192 & 230 & 219 & 232 & 300 & -76 & 67 & -53 & 40 & 4 & 46 & -18 & -107 \\ 160 & 275 & 150 & 135 & 244 & 294 & 267 & 331 & -2 & 90 & -17 & 10 & -24 & 49 & 29 & 89 \\ 153 & 189 & 113 & 173 & 260 & 342 & 256 & 176 & -20 & 18 & -38 & -4 & 24 & -75 & 25 & -5 \\ -12 & 7 & -9 & -13 & -6 & 11 & 12 & -69 & -10 & -1 & 14 & 6 & -38 & 3 & -45 & -99 \\ 10 & 3 & -31 & 16 & -1 & -51 & -10 & -30 & 2 & -12 & 0 & 24 & -32 & -45 & 109 & 42 \\ -7 & 5 & -44 & -35 & 67 & -10 & -17 & -15 & 3 & -15 & -28 & 0 & 41 & -30 & -18 & -19 \\ -4 & 9 & -1 & -37 & 41 & 6 & -33 & 2 & 9 & -12 & -67 & 31 & -7 & 3 & 2 & 0 \\ 2 & -3 & 9 & -25 & 2 & -25 & 60 & -8 & -11 & -4 & -123 & -12 & -6 & -4 & 14 & -12 \\ -1 & 22 & 32 & 46 & 10 & 48 & -11 & 20 & 19 & 32 & -59 & 9 & 70 & 50 & 16 & 73 \\ 43 & -18 & 32 & -40 & -13 & -23 & -37 & -61 & 8 & 22 & 2 & 13 & -12 & 43 & -8 & -45 \\ 16 & 2 & -6 & -32 & -7 & 5 & -13 & -50 & 24 & 7 & -61 & 2 & 11 & -33 & 43 & 1 \end{bmatrix}$$

237

到此我们完成了离散小波变换的一次变换。我们可以继续对变换结果的左上  $8 \times 8$  的部分  $I_{1,2}(x, y)$

继续做变换。结果如下:

$$I_{22}(x, y) = \begin{bmatrix} 558 & 451 & 608 & 532 & 75 & 26 & 94 & 25 & -33 & 6 & -15 & 5 & 24 & -29 & 38 & 120 \\ 463 & 511 & 627 & 566 & 66 & 68 & -43 & 68 & -45 & 11 & -2 & 9 & -31 & -26 & -74 & 23 \\ 464 & 401 & 478 & 416 & 14 & 84 & -97 & -229 & -70 & 43 & 56 & -23 & -41 & 21 & 82 & -81 \\ 422 & 335 & 477 & 553 & -88 & 46 & -31 & -6 & -52 & 27 & -14 & 23 & -2 & 90 & 49 & 12 \\ 14 & 33 & -56 & 42 & 22 & -43 & -36 & 1 & -41 & 14 & 31 & 23 & 57 & 60 & -78 & -3 \\ -13 & 36 & 54 & 52 & 12 & -21 & 51 & 70 & -76 & 67 & -53 & 40 & 4 & 46 & -18 & -107 \\ 25 & -20 & 25 & -7 & -35 & 35 & -56 & -55 & -2 & 90 & -17 & 10 & -24 & 49 & 29 & 89 \\ 46 & 37 & -51 & 51 & -44 & 26 & 39 & -74 & -20 & 18 & -38 & -4 & 24 & -75 & 25 & -5 \\ -12 & 7 & -9 & -13 & -6 & 11 & 12 & -69 & -10 & -1 & 14 & 6 & -38 & 3 & -45 & -99 \\ 10 & 3 & -31 & 16 & -1 & -51 & -10 & -30 & 2 & -12 & 0 & 24 & -32 & -45 & 109 & 42 \\ -7 & 5 & -44 & -35 & 67 & -10 & -17 & -15 & 3 & -15 & -28 & 0 & 41 & -30 & -18 & -19 \\ -4 & 9 & -1 & -37 & 41 & 6 & -33 & 2 & 9 & -12 & -67 & 31 & -7 & 3 & 2 & 0 \\ 2 & -3 & 9 & -25 & 2 & -25 & 60 & -8 & -11 & -4 & -123 & -12 & -6 & -4 & 14 & -12 \\ -1 & 22 & 32 & 46 & 10 & 48 & -11 & 20 & 19 & 32 & -59 & 9 & 70 & 50 & 16 & 73 \\ 43 & -18 & 32 & -40 & -13 & -23 & -37 & -61 & 8 & 22 & 2 & 13 & -12 & 43 & -8 & -45 \\ 16 & 2 & -6 & -32 & -7 & 5 & -13 & -50 & 24 & 7 & -61 & 2 & 11 & -33 & 43 & 1 \end{bmatrix}$$

$I_{1,2}(x, y)$  和  $I_{2,2}(x, y)$  的子带分别对应图 8-19 中的 a 和 b。现在, 我们可以对不同的子带使用不同级别的量化, 达到特定的效果和数据率。这就是简单小波变换压缩算法的基础。现在我们要说明小波变换的机理, 这里就不做量化, 而是直接重构函数。

我们把图 8-19 最左上角的  $8 \times 8$  图像开作为开始, 把第一列分成低通和高通两部分, 低通显然是数列前面的一半。在每个系数后面加一个 0, 得到如下结果:

$$\bar{a} = [558, 0, 463, 0, 464, 0, 422, 0]^T$$

$$\bar{b} = [14, 0, -13, 0, 25, 0, 46, 0]^T$$

因为这里用的是双正交滤波, 需要把  $\bar{a}$ 、 $\bar{b}$  分别和  $\tilde{h}_0[n]$ 、 $\tilde{h}_1[n]$  做卷积。将结果相加, 称为一个  $8 \times 1$  的数组, 如下所示:

$$[414, 354, 323, 338, 333, 294, 324, 260]^T$$

238 最左上角的  $8 \times 8$  图像每一列都这样处理之后, 得到如下结果:

$$I'_{21}(x, y) = \begin{bmatrix} 414 & 337 & 382 & 403 & 70 & -16 & 48 & 12 & -33 & 6 & -15 & 5 & 24 & -29 & 38 & 120 \\ 354 & 322 & 490 & 368 & 39 & 59 & 63 & 55 & -45 & 11 & -2 & 9 & -31 & -26 & -74 & 23 \\ 323 & 395 & 450 & 442 & 62 & 25 & -26 & 90 & -70 & 43 & 56 & -23 & -41 & 21 & 82 & -81 \\ 338 & 298 & 346 & 296 & 23 & 77 & -117 & -131 & -52 & 27 & -14 & 23 & -2 & 90 & 49 & 12 \\ 333 & 286 & 364 & 298 & 4 & 67 & -75 & -176 & -41 & 14 & 31 & 23 & 57 & 60 & -78 & -3 \\ 294 & 279 & 308 & 350 & -2 & 17 & 12 & -53 & -76 & 67 & -53 & 40 & 4 & 46 & -18 & -107 \\ 324 & 240 & 326 & 412 & -96 & 54 & -25 & -45 & -2 & 90 & -17 & 10 & -24 & 49 & 29 & 89 \\ 260 & 189 & 382 & 359 & -47 & 14 & -63 & 69 & -20 & 18 & -38 & -4 & 24 & -75 & 25 & -5 \\ -12 & 7 & -9 & -13 & -6 & 11 & 12 & -69 & -10 & -1 & 14 & 6 & -38 & 3 & -45 & -99 \\ 10 & 3 & -31 & 16 & -1 & -51 & -10 & -30 & 2 & -12 & 0 & 24 & -32 & -45 & 109 & 42 \\ -7 & 5 & -44 & -35 & 67 & -10 & -17 & -15 & 3 & -15 & -28 & 0 & 41 & -30 & -18 & -19 \\ -4 & 9 & -1 & -37 & 41 & 6 & -33 & 2 & 9 & -12 & -67 & 31 & -7 & 3 & 2 & 0 \\ 2 & -3 & 9 & -25 & 2 & -25 & 60 & -8 & -11 & -4 & -123 & -12 & -6 & -4 & 14 & -12 \\ -1 & 22 & 32 & 46 & 10 & 48 & -11 & 20 & 19 & 32 & -59 & 9 & 70 & 50 & 16 & 73 \\ 43 & -18 & 32 & -40 & -13 & -23 & -37 & -61 & 8 & 22 & 2 & 13 & -12 & 43 & -8 & -45 \\ 16 & 2 & -6 & -32 & -7 & 5 & -13 & -50 & 24 & 7 & -61 & 2 & 11 & -33 & 43 & 1 \end{bmatrix}$$

现在开始对行进行变换。对左上角的  $8 \times 8$  图像, 再把每一行分成低通和高通两部分, 在每个系数后面插入 0, 结果和  $\tilde{h}_0[n]$ 、 $\tilde{h}_1[n]$  做卷积。所有的行都进行相应处理, 可得到如下结果:

$$I'_{12}(x, y) = \begin{bmatrix} 353 & 212 & 251 & 272 & 281 & 234 & 308 & 289 & -33 & 6 & -15 & 5 & 24 & -29 & 38 & 120 \\ 280 & 203 & 254 & 250 & 402 & 269 & 297 & 207 & -45 & 11 & -2 & 9 & -31 & -26 & -74 & 23 \\ 269 & 202 & 312 & 280 & 316 & 353 & 337 & 227 & -70 & 43 & 56 & -23 & -41 & 21 & 82 & -81 \\ 256 & 217 & 247 & 155 & 236 & 328 & 114 & 283 & -52 & 27 & -14 & 23 & -2 & 90 & 49 & 12 \\ 240 & 221 & 226 & 172 & 264 & 294 & 113 & 330 & -41 & 14 & 31 & 23 & 57 & 60 & -78 & -3 \\ 206 & 204 & 201 & 192 & 230 & 219 & 232 & 300 & -76 & 67 & -53 & 40 & 4 & 46 & -18 & -107 \\ 160 & 275 & 150 & 135 & 244 & 294 & 267 & 331 & -2 & 90 & -17 & 10 & -24 & 49 & 29 & 89 \\ 153 & 189 & 113 & 173 & 260 & 342 & 256 & 176 & -20 & 18 & -38 & -4 & 24 & -75 & 25 & -5 \\ -12 & 7 & -9 & -13 & -6 & 11 & 12 & -69 & -10 & -1 & 14 & 6 & -38 & 3 & -45 & -99 \\ 10 & 3 & -31 & 16 & -1 & -51 & -10 & -30 & 2 & -12 & 0 & 24 & -32 & -45 & 109 & 42 \\ -7 & 5 & -44 & 35 & 67 & -10 & -17 & -15 & 3 & -15 & -28 & 0 & 41 & -30 & -18 & -19 \\ -4 & 9 & -1 & -37 & 41 & 6 & -33 & 2 & 9 & -12 & -67 & 31 & -7 & 3 & 2 & 0 \\ 2 & -3 & 9 & -25 & 2 & -25 & 60 & -8 & -11 & -4 & -123 & -12 & -6 & -4 & 14 & -12 \\ -1 & 22 & 32 & 46 & 10 & 48 & -11 & 20 & 19 & 32 & -59 & 9 & 70 & 50 & 16 & 73 \\ 43 & -18 & 32 & -40 & -13 & -23 & -37 & -61 & 8 & 22 & 2 & 13 & -12 & 43 & -8 & -45 \\ 16 & 2 & -6 & -32 & -7 & 5 & -13 & -50 & 24 & 7 & -61 & 2 & 11 & -33 & 43 & 1 \end{bmatrix}$$

之后用同样的方法, 对  $I'_{12}(x, y)$  进行处理, 得到  $I'_{00}(x, y)$ 。注意  $I'_{00}(x, y)$  和  $I_{00}(x, y)$  并不等同, 但差别不大。这些微小的差别是由正变换和逆变换过程, 以及计算过程中从浮点数转换成整数灰度值造成的。图 8-21 是一个 Haar 小波的三级分解。

239

$$I'_{00}(x, y) = \begin{bmatrix} 158 & 170 & 97 & 103 & 122 & 129 & 132 & 125 & 132 & 126 & 111 & 157 & 159 & 144 & 116 & 91 \\ 164 & 152 & 90 & 98 & 123 & 151 & 131 & 159 & 188 & 115 & 106 & 145 & 140 & 143 & 227 & 52 \\ 115 & 148 & 89 & 100 & 117 & 118 & 131 & 151 & 201 & 210 & 84 & 154 & 127 & 146 & 58 & 58 \\ 94 & 144 & 88 & 104 & 187 & 123 & 117 & 181 & 184 & 203 & 202 & 153 & 152 & 228 & 45 & 146 \\ 100 & 155 & 88 & 99 & 164 & 112 & 147 & 169 & 163 & 186 & 143 & 193 & 207 & 38 & 112 & 158 \\ 103 & 153 & 93 & 102 & 203 & 135 & 145 & 91 & 66 & 192 & 188 & 103 & 177 & 46 & 166 & 158 \\ 102 & 146 & 106 & 99 & 99 & 121 & 39 & 60 & 164 & 175 & 198 & 46 & 56 & 56 & 156 & 156 \\ 99 & 146 & 95 & 97 & 143 & 60 & 102 & 106 & 107 & 110 & 191 & 61 & 65 & 128 & 153 & 154 \\ 98 & 139 & 102 & 109 & 103 & 123 & 53 & 80 & 171 & 136 & 177 & 53 & 43 & 158 & 148 & 173 \\ 84 & 133 & 107 & 84 & 148 & 42 & 157 & 94 & 150 & 119 & 182 & 45 & 29 & 146 & 141 & 200 \\ 57 & 152 & 109 & 41 & 93 & 213 & 70 & 72 & 139 & 102 & 137 & 82 & 151 & 143 & 128 & 207 \\ 56 & 141 & 108 & 58 & 91 & 50 & 54 & 60 & 87 & 165 & 57 & 102 & 146 & 149 & 116 & 211 \\ 89 & 114 & 187 & 46 & 113 & 104 & 55 & 66 & 127 & 154 & 186 & 71 & 153 & 134 & 203 & 94 \\ 35 & 99 & 150 & 66 & 34 & 88 & 88 & 127 & 140 & 141 & 175 & 212 & 144 & 128 & 213 & 100 \\ 88 & 97 & 96 & 50 & 49 & 101 & 47 & 90 & 136 & 136 & 156 & 204 & 105 & 43 & 54 & 76 \\ 43 & 104 & 69 & 69 & 68 & 53 & 110 & 127 & 134 & 145 & 158 & 183 & 109 & 121 & 72 & 113 \end{bmatrix}$$

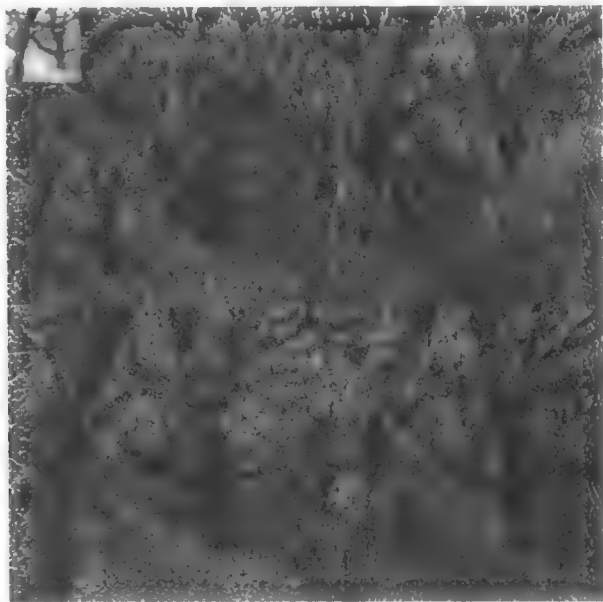


图 8-21 Haar 小波分解 (经 Steve Kiltbau 授权)

### 3. 小波约简程序

另一个基于小波变换的图像缩小算法更为简单，它的基本思想是仅仅保留最低频率。本书的网站上的 `wavelet_reduction.c` 是它的一个实例，仅限于调整缩放函数和分解滤波器，使得图像缩小若干倍。此程序支持 UNIX 下 PGM 文件格式，使用了表 8-3 中 Antonini 9/7 双正交滤波器。

## 8.7 小波包

小波包可以看作对小波的通用化。它是由 Coifman、Meyer、Quake 和 Wickerhauser[9]首先提出的，是系列标准正交基的离散函数  $R^N$ 。一个完整的子带分解可以看作作用深度为  $\log N$  的分析树对输入信号进行分解。

在通常的二元小波分解中，只有低通滤波带子带被递归分解，因而结果可以表示为对数树结构。然而，小波包分解可以用全树的任何子树表示。因此，分解之后的树形和所有允许的子波树同构[10]。任何一个子树的叶子表示一个可能的正交基。

小波包分解具有以下一些优点：

- 灵活性，因为在某种程度上寻找性能最优的基函数可以从大量的准入函数库中选取。

- 在频域和空域上都具有良好的定位性。
- 计算复杂度小, 因为任一分解都可以用快速滤波器库完成, 复杂度为  $N \log N$ 。

小波包目前应用广泛, 例如图像压缩、信号去噪处理、指纹检测等。

## 8.8 小波系数的嵌入零树

到目前为止, 我们已经介绍了用于图像分解的基于小波的方案。然而, 除了提到量化掉小系数的思想, 我们还没有真正解决如何对小波变换值进行编码, 即怎样形成位流。这个问题的处理需要采用一种新的数据结构——嵌入零树。

嵌入式零树小波 (Embedded Zerotree Wavelet, EZW) 算法是由 Shapiro[11]提出的, 它是图像编码中一种有效且计算效率高的技术。此后又有许多人对最初的 EZW 算法做了改进, 其中最值得关注的是 Said 和 Pearlman 的层次树集合划分 (Set Partitioning in Hierarchical Trees, SPIHT) 算法[12]和 Taubman 的优化截断嵌入式块编码 (Embedded Block Coding With Optimized Truncation, EBCOT) 算法[13], 后者被 JPEG2000 标准所采用。

EZW 算法解决两个问题: 在给定的码率下获得最优的图像质量并以一种嵌入式的方式完成这项任务。嵌入 (embedded) 码是一种将所有低率码包含在位流的开始部分的编码。每个位在位流中按重要性有效地排列。嵌入码允许编码器在任意点结束编码, 从而准确地满足任何目标码率。同样, 解码器也可以在任意点结束解码, 产生和低率编码对应的重现值。

为了达到这一目标, EZW 算法利用了低码率图像编码的一个重要优点。用常规的编码方法实现低码率时, 要使用标量量化加上熵编码, 这样, 在量化之后最可能的符号是零。其结果是大部分的位都用于对重要性图的编码, 重要性图标记输入样本 (在二维离散小波变换的情形下为变换系数) 的量化值是零值还是非零值。EZW 算法正是利用了这一点, 将对重要性图的编码中的任一显著改进转化为相应的压缩效率的提高。EZW 算法由两个中心部分组成: 零树数据结构和逐次近似量化方法。

### 8.8.1 零树数据结构

我们使用一种称为零树 (zerotree) 的新数据结构实现对重要性图的编码。对于小波系数  $x$  和给定的阈值  $T$ , 如果  $|x| < T$ , 我们称小波系数  $x$  不重要 (insignificant)。如果下述假设成立, 则零树起作用: 如果在粗 (大) 尺度下的一个小波系数对于给定的阈值  $T$  不重要, 那么细 (小) 尺度下相同空间位置相同方向的所有的小波系数对于  $T$  都可能不重要。使用第 8 章介绍的层次小波分解, 就可以将一个给定尺度下的每一个系数同相同方向上下一个更高缩放比例下的一组系数联系起来。

图 8-22 显示了三级小波分解的零树。粗 (大) 尺度下的系数被称作父 (parent) 节点, 而下一级细 (小) 尺度下相同空间位置相同方向的系数称作子 (children) 节点。对某一确定的父节点, 所有比其更精细的缩放比例下的所有系数的集合称为后代 (descendant)。同样对某一确定的子节点, 所有比其更粗的尺度下的所有系数的集合称为祖先 (ancestor)。

对系数的扫描遵循这样的原则, 即子节点不能在其父节点之前被扫描。图 8-23 描述了一个三级小波分解的扫描方式。

对一个给定的阈值  $T$ , 如果系数  $x$  不重要, 并且它的所有后代也都不重要, 就称该系数是零树的一个元素。零树中的元素如果不是上一个零树根的后代, 那么它就是一个零树根。可以用四符号的零树对重要性图进行编码, 这四个符号是:



- **零树根** 零树根用一种特殊的符号编码, 这个符号表明更高缩放比例下该系数不重要是完全可预知的。
- **独立零** 该系数不重要, 但有重要的后代。

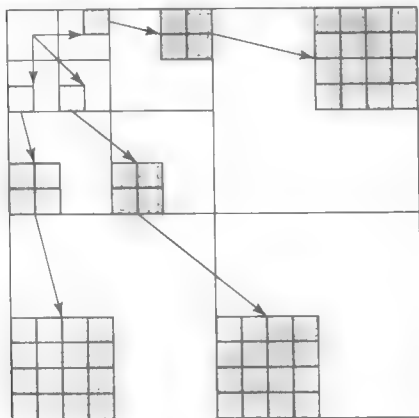


图 8-22 零树中的父子关系

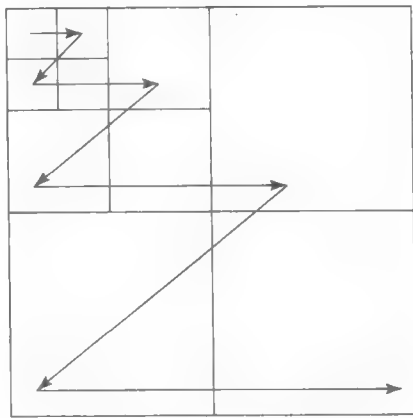


图 8-23 EZW 的扫描次序

- **正重要性** 该系数重要, 并且是正值。
- **负重要性** 该系数重要, 并且是负值。

采用零树后, 大大减少了对重要性图的编码成本。零树的操作利用了变换系数的自相似性。采用零树最重要的理由就在于即使图像经过了解相关的变换, 出现不重要的系数也不是独立事件。

另外, 零树编码技术基于这样一个经验, 即预测不同尺度下的不重要性比预测重要性容易得多。这一技术主要关注减小对重要性图编码的成本, 从而可以采用更多的位来对耗用位更多的有效系数进行编码。

### 8.8.2 逐次近似量化

EZW 编码器中的嵌入式编码通过采用一种称为逐次近似量化 (Successive Approximation Quantization, SAQ) 的方法来进行。采用这种方法的一个目标是生成嵌入式的编码, 从而为小波变换图像对应的缩放比例空间提供从粗略到精细的多精度的对数表示。另一个目标是进一步利用零树数据结构在重要性图编码上的高效率, 对更多的重要性图进行编码。

SAQ 方法顺序地应用一个阈值序列  $T_0, \dots, T_{N-1}$  来判定每个系数的重要性, 阈值的选择使得  $T_i = T_{i-1}/2$ 。第一个阈值  $T_0$  应满足对所有的变换系数  $x_j$  都有  $|x_j| < 2T_0$ 。在编码和解码过程中生成主表和副表, 主表中包含还没有被判定是重要性的系数的坐标, 其相对顺序与初次扫描相同。

按照图 8-23 所示的扫描顺序, 某一特定子带的所有系数比其下一个子带的系数优先出现在初始主表中。副表所包含的是那些被判定为重要的系数的值。对每个阈值, 每个表都只扫描一次。

在主扫描过程中, 系数的坐标在主表中表明它还是不重要的。这些系数会与阈值  $T_i$  比较以判断其重要性。如果一个系数被判断为重要的, 那么它的幅值就被添加到副表中, 同时小波变换数组的系数置为零, 这样在以后的主扫描中对于更小的阈值仍然可以生成零树。最后得到的重要性图就已经是经过零树编码的了。

主扫描之后是副扫描, 副表中的所有系数及其幅值都要被扫描, 在解码器端完成这一步后, 就可以将其幅度的精度位数提高一位; 其结果是系数真实幅值的不确定度区间的宽度减小一半。

对副表中的每个幅值,这一改进可以用二值的字符表来编码,其中 1 表示真实值落在不确定度区间的上半部分,而 0 表示落在下半部分。用上面得到的二值字符串就可以进行熵编码。在副扫描之后,副表中的幅值就以降序排列,直到解码器也可以进行相同的排序。

两种扫描不断交替进行,不过每次主扫描之前,阈值都要减半。当满足某一目标停止条件时,编码过程才停下来。

### 8.8.3 EZW 示例

下面的例子说明了零树编码和逐次近似量化的概念。Shapiro[11]在其论文中给出了一个  $8 \times 8$  三级小波变换的 EZW 编码的例子。与 Shapiro 所给的例子不同,我们将完成编码和解码过程,并且给出熵编码之前的输出位流。

图 8-24 所示为我们试图采用 EZW 算法进行编码的三级小波变换的系数。我们用符号  $p$ 、 $n$ 、 $t$  和  $z$  来分别表示正重要性、负重要性、零树根和孤立零值。

由于最大的系数是 57,因此我们选择初始阈值  $T_0$  为 32。开始,主表包含所有系数的坐标。按图 8-23 所示的顺序开始扫描,判断各个系数的重要性,以下是按扫描顺序依次访问到的系数:

$$\{57, -37, -29, 30, 39, -20, 17, 33, 14, 6, 10, 19, 3, 7, 8, 2, 2, 3, 12, -9, 33, 20, 2, 4\}$$

对于阈值  $T_0 = 32$ ,很容易看出系数 57 和 -37 是重要的。这样,输出一个  $p$  和一个  $n$  来表示它们。系数 -29 本身不重要,不过它包含了一个重要的后代,即 LH1 中的 33,因此把它编码为  $z$ 。系数 30 也是不重要的,并且它所有的后代对于当前的阈值来说都是不重要的,所以将其编码为  $t$ 。

由于前面已经判定了 30 及其后代的不重要性,因此以后的扫描将绕过它们,也不会再产生多余的符号。按这种方式进行下去,主扫描输出如下的符号:

$$D_0: pnztpptptztppttt$$

有五个系数被判定为重要的,即 57、-37、39、33 和另一个 33。显然,没有一个系数大于  $2T_0 = 64$ ,而第一次主扫描用的阈值是 32,因此不确定度区间就是  $[32, 64]$ ,重要性系数就在这个不确定度区间的某处。

主扫描之后的副扫描指出系数落在不确定度区间的前半部分还是后半部分,从而改进这些系数的幅值。如果系数值落在区间  $[32, 48)$ ,输入为 0,落在区间  $[48, 64)$  则为 1。按照扫描的顺序,副扫描输出下面的位:

$$S_0: 10000$$

这时主表中包含除了被判定为重要的所有系数的坐标,而副表包含的值为  $\{57, 37, 39, 33, 33\}$ 。副扫描结束后,重新排列副表中的数值,使得较大的系数在较小系数的前面,排列还要符合一个约束,即解码器也可以进行相同的排列。

由于副扫描将不确定度区间分成两半,解码器就能够分辨区间  $[32, 48)$  和  $[48, 64)$  内的数值。39 和 37 在解码器端无法区分,因而它们的次序也不会改变。因此,在重新排序过程之后,副表仍旧保持原样。

在开始第二轮的主扫描和副扫描之前,应该将小波变换数组里重要性系数的值置为 0,这样就不会影响新的零树的产生。

57	-37	39	-20	3	7	9	10
-29	30	17	33	8	2	1	6
14	6	15	13	9	-4	2	3
10	19	-7	9	-7	14	12	-9
12	15	33	20	-2	3	1	0
0	7	2	4	4	-1	1	1
4	1	10	3	2	0	1	0
5	6	0	0	3	1	2	1

图 8-24 一个作为 EZW 算法输入的三级小波变换的系数

243  
244

245



## 8.9 层次树的集合划分

层次树的集合划分 (Set Partitioning in Hierarchical Trees, SPIHT) 是 EZW 算法的一种革命性扩展。基于 EZW 的基本原理, 包括对变换后系数进行部分排序, 细分位的有序位平面传输和利用变换后的小波图像的自相似性, SPIHT 算法通过改变系数子集划分和细分信息传送的方法, 大大地提高了处理器的性能。

SPIHT 位流的一个独特的性质就是它的紧密性。SPIHT 算法的结果位流紧凑到其通过熵编码器时, 即使经过大量的计算也只能产生边际压缩增益。因此, 没有熵编码器或者只有一个简单的无专利的赫夫曼编码器, 也可以实现快速的 SPIHT 编码器。

247

SPIHT 算法的另一个特征是不用给解码器传输排序信息。解码器会重新生成编码器的执行路径, 从而恢复排序信息。这样做的一个好处是编码器和解码器的执行时间差不多, 这一点是其他编码方法很难实现的。Said 和 Pearlman[12]对这种算法给出了详细的介绍。

## 8.10 进一步探索

Sayood[14]在以组织良好和易懂的方式进行有损压缩方面做了大量研究。

Gersho 和 Gray[15]对量化, 尤其是矢量量化进行了全面的探讨。除了基本理论之外, 该书还全面介绍了已有的 VQ 方法。

Gonzales 和 Woods[7]讨论的是数学变换和图像压缩, 对图像处理领域的算法进行了浅显易懂的解释。

很多有损压缩的数学基础是随机过程 (stochastic process), Stark 和 Woods[16]是介绍这一主题的很好的教材。

最后, Mallat[5]是一本关于小波的书, 它比较偏重理论。

本书网站上的本章的 Further Exploration 目录下的链接有:

- 一个在线的基于图形的小波变换演示。其中包括两个程序, 一个是关于一维小波变换的演示, 另一个是关于二维小波变换的演示。在一维变换的演示程序中, 你可以任意画出要变换的曲线。
- Theory of Data Compression (图像压缩理论) 网站, 内容包括有损数据压缩和无损数据压缩的基本理论。从这里还可以下载香农在 1948 年发表的论文。
- comp.compression 和 comp.compression.research 新闻组的 FAQ。它回答了大部分关于小波理论和数据压缩的常见问题。
- 一系列关于标量量化和矢量量化的幻灯片, 这些幻灯片来自 Delft 大学开设的信息理论课程。
- 一篇很优秀的文章 *Image Compression — from DCT to Wavelets: A Review*。
- 一些关于量化的资料和源代码。

248

## 8.11 练习

1. 假设现有一个无界的数据源, 我们希望使用  $M$  位的中宽型均匀量化器来量化它, 请给出步长为 1 时的总失真量。
2. 假设一个均匀量化器的域为  $[-b_M, b_M]$ , 我们定义其装载分数为

$$\gamma = \frac{b_M}{\sigma}$$

其中  $\sigma$  为源的标准差。写一段用 4 位均匀量化器量化一个高斯分布源的简单程序, 高斯分布的

均值为 0，方差为一个单位。图示出此装载分数的 SNR，并从图上估计使产生的失真为最小的最优步长值。

3. \*假设输入源服从均值为 0，方差为一个单位的高斯分布，也就是其概率密度函数为

$$f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \quad (8.66)$$

我们希望找到一个 4 级 Lloyd-Max 量化器。令  $y_i = [y_i^0, \dots, y_i^3]$ ， $b_i = [b_i^0, \dots, b_i^3]$ ，初始重现级设置为  $y_0 = [-2, -1, 1, 2]$ ，源是无界的，因而其两个外边界为  $+\infty$  和  $-\infty$ 。

按照本章介绍的 Lloyd-Max 算法，其他边界值为重现值的中点，这样  $b_0 = [-\infty, -1.5, 0, 1.5, +\infty]$ 。

使用式 (8.13) 对  $i=1$  再进行一次迭代，利用数值积分得出  $y_0^1$ 、 $y_1^1$ 、 $y_2^1$ 、 $y_3^1$ ，并且计算出  $y_1$  和  $y_0$  的方差。

迭代过程继续进行，直到相邻重现级的估计值的方差低于某一设定的阈值  $\epsilon$ 。写一段程序来实现如上所述的 Lloyd-Max 量化器。

4. 如果一个二维 DCT 的块大小为  $8 \times 8$ ，只用其 DC 分量来产生一个草图，原始像素的哪些部分会被使用到？

5. 当块的大小为 8 时，DCT 的定义由式 (8.17) 给出。

249

(a) 对于灰度范围是 0~255 的  $8 \times 8$  图片，DCT 系数的最大值可能是多少，这种情况发生在那张输入图片下？（给出这种图片所有的 DCT 系数。）

(b) 如果对整张图片都减去一个固定值 128，再进行 DCT，会对 DCT 值  $F[2.3]$  有什么样的影响？

(c) 为什么要进行这一减法运算？这样做会影响到对图像编码所需的位数吗？

(d) 在 IDCT 中有没有可能将这一减法运算转化回来？如果可能的话，应怎么做？

6. 利用三维 DCT 可以在视频流中得到相似的 DCT 方案。假设视频中的色彩分量是时刻  $k$  在位置  $(i, j)$  的像素  $f_{ijk}$ ，如何定义三维 DCT 变换？

7. 假设有一个光源照射下的颜色均匀的球，其阴影沿着球的表面光滑变化，如图 8-27 所示。

(a) 这张图片的 DCT 系数会是什么样子？

(b) 球的表面如果是棋盘状的颜色，其 DCT 系数又会是什么样子？

(c) 对于颜色均匀的球，请描述横跨球的上边缘与黑色背景接合处的块的 DCT 值。

(d) 请描述一下横跨球的左边缘的块的 DCT 值。

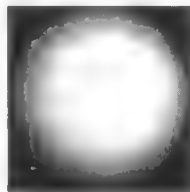


图 8-27 单光源照射下的球

8. Haar 小波的缩放函数定义如下

$$\phi(t) = \begin{cases} 1 & (0 \leq t \leq 1) \\ 0 & (\text{其他}) \end{cases} \quad (8.67)$$

其缩放向量为  $h_0[0] = h_0[1] = 1/\sqrt{2}$ 。

(a) 请画出缩放函数，并证明其平移  $\phi(2t)$  和  $\phi(2t-1)$  满足扩展方程 (8.56)。由这些函数的组合，得出完整的函数  $\phi(t)$ 。

(b) 从式 (8.59) 推导出小波向量  $h_1[0]$ 、 $h_1[1]$ ，然后由式 (8.57) 推导出 Haar 小波函数  $\psi(t)$ 。

250

9. 假设母函数  $\psi(t)$  的削减矩  $M_p$  不大于  $M_n$ ，将  $f(t)$  在  $t=0$  处的泰勒级数展开到  $f$  的  $n$  阶导数（也就是展开到剩余误差为  $O(n+1)$  阶）。求将上面的泰勒级数替换到式 (8.52) 产生的积分和，并且证明结果的阶数为  $O(s^{n+2})$ 。

10. 本书的网站上的程序 `wavelet_compression.c` 实际上是一个 MATLAB 函数（或者类似的第四代语言）。这样做的优点在于 `imread` 函数可以读入许多种图像格式，`imwrite` 函数可以输出所需的类型。以所给的程序为模板，编写一个用于小波图像还原的 MATLAB 程序，有可能把小波的数量作为一个函数参数。
11. 使用 MAPLE 这样的符号操作系统对函数进行傅里叶变换将非常简单。在 MAPLE 中，只需调用 `fourier` 函数，就可以直接看到结果。作为一个例子，试着输入下面的代码：

```
with('inttrans');
f := 1;
F := fourier(f,t,w);
```

答案应该是  $2\pi\delta(w)$ 。再试一个高斯型的：

```
f := exp(-t^2);
F := fourier(f,t,w);
```

这次的结果应该是  $\sqrt{\pi}e^{-w^2/4}$ ，可见高斯型的傅里叶变换是另一个高斯型。

12. 定义小波函数为

$$\psi(t) = \exp(-t^{1/4})\sin(t^4), \quad t \geq 0 \quad (8.68)$$

函数值在 0 附近波动。利用画图软件证明对于任意的  $p$  值，该函数的零矩  $M_p$  都为 0。

13. 分别实现基于 DCT 和基于小波的图像编码器。设计用户界面，使得两种编码器的压缩结果可以并排地显示出来，以便进行比较。为了进行数值上的比较，每幅压缩图片的 PSNR 也应当被显示出来。

另外再加入一个滑动条，来控制两个编码器的目标码率。若改变目标码率，各个编码器就即时地压缩输入图像，并将压缩结果立刻显示在用户界面上。

从质量上和数值上比较目标码率为 4bpp、1bpp 和 0.25bpp 时程序的压缩结果。

14. 写一个简单的程序或者参考本书的网站上的示例 DCT 程序 `dct_1D.c`，验证本章一维 DCT 的例 8-2 的结果。

251

## 8.12 参考文献

- [1] A. György, "On the Theoretical Limits of Lossy Source Coding," Tudományos Diákkör (TDK) Conference [Hungarian Scientific Student's Conference] at Technical University of Budapest, 1998. <http://www.szit.bme.hu/~gya/mixed.ps>.
- [2] S. Arimoto, "An Algorithm for Calculating the Capacity of an Arbitrary Discrete Memoryless Channel," *IEEE Transactions on Information Theory*, 18: 14–20, 1972.
- [3] R. Blahut, "Computation of Channel Capacity and Rate-Distortion Functions," *IEEE Transactions on Information Theory*, 18: 460–473, 1972.
- [4] J.F. Blinn, "What's the Deal with the DCT?" *IEEE Computer Graphics and Applications*, 13(4): 78–83, 1993.
- [5] S. Mallat, *A Wavelet Tour of Signal Processing*, San Diego: Academic Press, 1998.
- [6] S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11: 674–693, 1989.
- [7] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2002.
- [8] B.E. Usevitch, "A Tutorial on Modern Lossy Wavelet Image Compression: Foundations of JPEG 2000," *IEEE Signal Processing Magazine*, 18(5): 22–35, 2001.
- [9] R. Coifman, Y. Meyer, S. Quake, and V. Wickerhauser, "Signal Processing and Compression with Wavelet Packets," Numerical Algorithms Research Group, Yale University, 1990.

- [10] K. Ramachandran and M. Vetterli, "Best Wavelet Packet Basis in a Rate-Distortion Sense," *IEEE Transactions on Image Processing*, 2: 160–173, 1993.
- [11] J. Shapiro, "Embedded Image Coding using Zerotrees of Wavelet Coefficients," *IEEE Transactions on Signal Processing*, 41(12): 3445–3462, 1993.
- [12] A. Said and W.A. Pearlman, "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3): 243–250, 1996.
- [13] D. Taubman, "High Performance Scalable Image Compression with EBCOT," *IEEE Transactions on Image Processing*, 9(7): 1158–1170, 2000.
- [14] K. Sayood, *Introduction to Data Compression*, 2nd ed., San Francisco: Morgan Kaufmann, 2000.
- [15] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*, Boston: Kluwer Academic Publishers, 1992.
- [16] H. Stark and J.W. Woods, *Probability and Random Processes with Application to Signal Processing*, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 2001.

## 第9章 图像压缩标准

近几年以来,由于数字图像设备(例如扫描仪、数码相机等)的飞速发展,数字图像的应用需求大规模增加。如何以数字方式有效地处理和存储这些图像?这些需求大大促进了图像压缩标准的发展。一般说来,标准比特定的程序或设备有更长的生命力,更值得我们认真学习研究。在这一章里,我们将介绍现有的一些标准,同时向大家说明如何将第7章和第8章中介绍的知识应用到实际中。

我们将首先讨论在 Web 上应用最广的标准 JPEG 定义,然后是基于小波的 JPEG2000 标准。为了完整起见,我们还将介绍另外的两个标准, JPEG-LS (一种无损的 JPEG) 和 JBIG (应用于二值图像压缩的标准)。

### 9.1 JPEG 标准

JPEG 是由联合图像专家组 (Joint Photographic Experts Group) 开发的一种图像压缩标准。1992 年,它被正式采纳为国际标准[1]。

JPEG 由很多步骤组成,每一步都对压缩有贡献。我们将首先介绍采取每一步的动机,然后讨论所用到的各种算法。

#### 9.1.1 JPEG 图像压缩的主要步骤

我们已经知道,数字图像  $f(i, j)$  并不像一维音频信号一样定义在时间域上,它定义在空间域 (spatial domain) 上。图像是两维变量  $i$  和  $j$  的函数 (或者表示为  $x, y$ )。2D DCT 作为 JPEG 中,的一步得到空间频率域的频率响应,用函数  $F(u, v)$  表示。

JPEG 是有损的图像压缩方法。在 JPEG 中, DCT 变换的编码效率基于下述 3 个特性:

**特性 1** 在图像区域内,有用的图像内容变化相对缓慢,也就是说,在一个小区域内 (例如,在一个  $8 \times 8$  的图像块内) 亮度值的变换不会太频繁。空间频率表示在一个图像块内像素值的变化次数。DCT 形式化地表明了这一变化,它把对图像内容的变化度量和每一个块的余弦波周期数对应起来。

**特性 2** 心理学实验表明,在空间域内,人类对高频分量损失的感知能力远远低于对低频分量损失的感知能力。

在 JPEG 中应用 DCT 主要是为了在减少高频内容,同时更有效地将结果编码为位串。我们用空间冗余 (spatial redundancy) 来说明在一张图像里很多信息是重复的。例如,如果一个像素是红色,那么与它临近的像素也很有可能是红色。正如特性 2 所表明的, DCT 的低频分量系数非常重要。因此,随着频率的增加,准确表示 DCT 系数的重要性随之降低。我们甚至可以把它置零,也不会感知到很多信息的丢失。

显然,一个零串可以表示为零的长度数,用这种方式,比特的压缩成为可能。我们可以用很少的数来表示一个块中的像素,摒弃一些和位置相关的信息,达到摒弃空间冗余的目的。

JPEG 可用于彩色和灰度图像。在彩色图像的情况下 (如在 YIQ 或 YUV 中),编码器在各自的分量上工作,但使用相同的例程。如果源图像是其他的格式,编码器会执行颜色空间转换,将其转换为 YIQ 或 YUV 空间。正如在第 5 章里讨论的,色度图像 ( $I, Q$  或  $U, V$ ) 是二次采样 (subsample) 的: JPEG 采用  $4:2:0$  的方案,利用视觉的下面一个特性。



**特性 3** 人类对灰度（黑和白）的视觉敏感度（区分相近空间线的准确度）要远远高于对彩色的敏感度。如果彩色发生很接近的变化，那么我们很难区分出来（设想在漫画中使用的斑点状的墨水）。这是因为我们的眼睛对黑色的线最敏感，并且我们的大脑总是把这种颜色推广开来。实际上，普通的电视就是利用这个原理，总是传播较多的灰度信息，较少的颜色信息。

当我们浏览 JPEG 图片时，3 个被压缩的分量独立地被解码然后合并起来。在颜色通道，每一个像素首先被扩大为一个  $2 \times 2$  的块。为了满足普遍性，我们选用其中的一个分量（例如，Y 图像）来描述压缩算法。

图 9-1 给出了一个 JPEG 编码器的示意图。如果我们逆着箭头，就得到一个 JPEG 的解码器。JPEG 的编码包括以下一些主要步骤：

- 把 RGB 转换为 YIQ 或 YUV，并且二次采样。
- 对图像块进行 DCT 变换。
- 进行量化。
- 进行 Z 编序和游长编码。
- 进行熵编码。

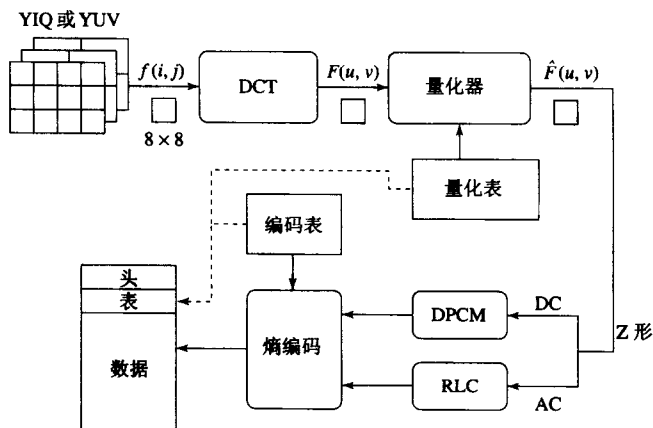


图 9-1 JPEG 编码器的模块图

### 1. 图像块的 DCT 变换

每一个图像被划分为  $8 \times 8$  的块。将 2D DCT 变换（式（8.17））应用到每一个块图像  $f(i, j)$ ，得到相应的输出函数——每个块的 DCT 变换的系数  $F(u, v)$ 。JPEG 中图像块大小的选择是由委员经过仔细权衡后确定的，因为大于 8 的数值会使图像在低频处的效果更好，但使用 8 能使 DCT（及 ICT）变换的计算速度更快。

使用“块”这种方式，会使得某个块与临近的块隔离开来。这就是为什么当我们采用比较高的压缩率时，JPEG 图像看起来不够连贯，甚至能看到一个个的块（事实上消除这种现象是研究者的一个重要工作）。

要计算一个特定的  $F(u, v)$ ，我们选择图 8-9 的基本图像和相应的  $u, v$ ，并且利用式（8.17）推导频率响应函数  $F(u, v)$ 。

### 2. 量化

JPEG 中量化的目标是减少压缩图像所需要的位数[2]。它由每个频率除以一个整数，然后取整得到：

$$\hat{F}(u,v) = \text{round} \left( \frac{F(u,v)}{Q(u,v)} \right) \tag{9.1}$$

其中， $F(u,v)$  表示 DCT 系数， $Q(u,v)$  是量化矩阵 (quantization matrix)， $\hat{F}(u,v)$  表示量化后的 DCT 系数，JPEG 将会在后面的熵编码中使用到这些矩阵。

255

适用于亮度和色度图的  $8 \times 8$  量化矩阵  $Q(u,v)$  中的默认值见表 9-1 和表 9-2。这些值是从心理学的研究结果中得来的，能够得到最大的压缩率，同时能使 JPEG 图片的感知损失最小。接下来的结论就显然易见了：

- 由于  $Q(u,v)$  中的值都相对较大，所以  $\hat{F}(u,v)$  的值的大小和变化都远远小于  $F(u,v)$ 。在后面我们将会看到  $\hat{F}(u,v)$  能用很少的位编码。量化是 JPEG 压缩中产生信息丢失的主要原因。
- 每一个  $Q(u,v)$  表都在右下角置较大的值。这样做的目的在于丢弃更多的高频分量，特性 1 和特性 2 都说明了这样做的好处。

表 9-1 亮度量化表

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

表 9-2 色度量化表

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

我们可以给  $Q(u,v)$  矩阵乘以比例值来改变压缩率。实际上，每一个 JPEG 应用提供给用户的质量因子 (quality factor)，本质上都是和缩放因子线性联系在一起的。JPEG 也允许指定定制量化表并放在头部。我们可以设定特殊的值，例如  $Q \equiv 2$  或  $Q \equiv 100$ ，来观察  $Q$  值的改变所带来的直观影响。

256

图 9-2 和 9-3 表明了对测试图像 Lena 进行 JPEG 编码和解码的结果。图中仅显示亮度图 (Y) 的结果。量化后的无损压缩步骤就不再赘述，因为它们对 JPEG 的图像质量没有影响。结果说明了对相对平滑和相对粗糙的块分别进行压缩和解压后的结果。

假设  $f(i,j)$  代表从图像中截取的一个  $8 \times 8$  的块， $F(u,v)$  是 DCT 变换系数， $\hat{F}(u,v)$  是量化后的 DCT 系数。 $\tilde{F}(u,v)$  表示未量化的 DCT 系数，乘以  $Q(u,v)$  得到， $\tilde{f}(i,j)$  是重现后得到的图像块。在图 9-2 和图 9-3 最后一行的误差表明了 JPEG 压缩的质量，特别是压缩后的损失。

在图 9-2 中，选择一个亮度变化平滑的图像块 (图中用黑框表示)。块的左边稍亮，右边稍暗，正如所期望的那样，除了 DC 和前几个 AC 分量 (用来表示空间低频分量)，大部分的 DCT 系数  $F(u,v)$  的幅值都很小。这是因为在这一个块中的像素值所含空间高频分量的改变较小的缘故。

另一个缩小处理规模的办法是整齐化。8 位亮度值  $f(i,j)$  的取值范围是  $[0, 255]$ 。在 JPEG 的实现中，每一个  $Y$  值首先都被减去 128。

这里的主要思想是使  $Y$  分量成为 0 均值图像，这和色度图相类似。结果，我们并没有在为均值编码的过程中损失任何的位 (例如一个取值范围为 120~135 的  $8 \times 8$  块)。0 均值处理后的图像不会影响到输出中的 AC 系数，它改变的只是 DC 系数。



Lena Y 图像中的一个 8×8 的块

200 202 189 188 189 175 175 175	515 65 -12 4 1 2 -8 5
200 203 198 188 189 182 178 175	-16 3 2 0 0 -11 -2 3
203 200 200 195 200 187 185 175	-12 6 11 -1 3 0 1 -2
200 200 200 200 197 187 187 187	-8 3 -4 2 -2 -3 -5 -2
200 205 200 200 195 188 187 175	0 -2 7 -5 4 0 -1 -4
200 200 200 200 200 190 187 175	0 -3 -1 0 4 1 -1 0
205 200 199 200 191 187 187 175	3 -2 -3 3 3 -1 -1 -3
210 200 200 200 188 185 187 186	-2 5 -2 4 -2 2 -3 0
$f(i, j)$	$F(u, v)$
32 6 -1 0 0 0 0 0	512 66 -10 0 0 0 0 0
-1 0 0 0 0 0 0 0	-12 0 0 0 0 0 0 0
-1 0 1 0 0 0 0 0	-14 0 16 0 0 0 0 0
-1 0 0 0 0 0 0 0	-14 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
$\hat{F}(u, v)$	$\tilde{F}(u, v)$
199 196 191 186 182 178 177 176	1 6 -2 2 7 -3 -2 -1
201 199 196 192 188 183 180 178	-1 4 2 -4 1 -1 -2 -3
203 203 202 200 195 189 183 180	0 -3 -2 -5 5 -2 2 -5
202 203 204 203 198 191 183 179	-2 -3 -4 -3 -1 -4 4 8
200 201 202 201 196 189 182 177	0 4 -2 -1 -1 -1 5 -2
200 200 199 197 192 186 181 177	0 0 1 3 8 4 6 -2
204 202 199 195 190 186 183 181	1 -2 0 5 1 1 4 -6
207 204 200 194 190 187 185 184	3 -4 0 6 -2 -2 2 2
$\tilde{f}(i, j)$	$\epsilon(i, j) = f(i, j) - \tilde{f}(i, j)$

图 9-2 平滑图像块的 JPEG 压缩

在图 9-3 中，我们选择了一个亮度变化剧烈的图像块。AC 分量具有很大的幅值（尤其是  $u$ ， $v$  值都很大的右下角）。可以看出，此时的误差  $q(i, j)$  大于图 9-2——如果图像变化较为剧烈，JPEG 会引起较大的损失。

### 3. 熵编码的准备

上面我们介绍了 JPEG 压缩的两个主要步骤：DCT 变换和量化。正如图 9-1 的流程图所示，下一步工作是对量化后的 DCT 系数进行熵编码（entropy coding）。这一步不会造成数据的损失。不过，在熵编码步骤的准备过程中，DC 系数和 AC 系数的处理是不同的。AC 分量进行游长编码，DC 分量采用 DPCM 编码。



Lena Y 图像中的另一个 8×8 的块

70 70 100 70 87 87 150 187	-80 -40 89 -73 44 32 53 -3
85 100 96 79 87 154 87 113	-135 -59 -26 6 14 -3 -13 -28
100 85 116 79 70 87 86 196	47 -76 66 -3 -108 -78 33 59
136 69 87 200 79 71 117 96	-2 10 -18 0 33 11 -21 1
161 70 87 200 103 71 96 113	-1 -9 -22 8 32 65 -36 -1
161 123 147 133 113 113 85 161	5 -20 28 -46 3 24 -30 24
146 147 175 100 103 103 163 187	6 -20 37 -28 12 -35 33 17
156 146 189 70 113 161 163 197	-5 -23 33 -30 17 -5 -4 20
$f(i, j)$	$F(u, v)$
-5 -4 9 -5 2 1 1 0	-80 -44 90 -80 48 40 51 0
-11 -5 -2 0 1 0 0 -1	-132 -60 -28 0 26 0 0 -55
3 -6 4 0 -3 -1 0 1	42 -78 64 0 -120 -57 0 56
0 1 -1 0 1 0 0 0	0 17 -22 0 51 0 0 0
0 0 -1 0 0 1 0 0	0 0 -37 0 0 109 0 0
0 -1 1 -1 0 0 0 0	0 -35 55 -64 0 0 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
$\hat{F}(u, v)$	$\tilde{F}(u, v)$
70 60 106 94 62 103 146 176	0 10 -6 -24 25 -16 4 11
85 101 85 75 102 127 93 144	0 -1 11 4 -15 27 -6 -31
98 99 92 102 74 98 89 167	2 -14 24 -23 -4 -11 -3 29
132 53 111 180 55 70 106 145	4 16 -24 20 24 1 11 -49
173 57 114 207 111 89 84 90	-12 13 -27 -7 -8 -18 12 23
164 123 131 135 133 92 85 162	-3 0 16 -2 -20 21 0 -1
141 159 169 73 106 101 149 224	5 -12 6 27 -3 -2 14 -37
150 141 195 79 107 147 210 153	6 5 -6 -9 6 14 -47 44
$\tilde{f}(i, j)$	$\epsilon(i, j) = f(i, j) - \tilde{f}(i, j)$

图 9-3 纹理图像块的 JPEG 压缩

#### 4. AC 系数的游长编码

大家会注意到在图 9-2 中, 应用量化后,  $\hat{F}(u, v)$  中出现了很多零。游长编码 (Run-Length Coding, RLC; 或 Run-Length Encoding, RLE) 可以把  $\hat{F}(u, v)$  的值变换为集合  $\{\# \text{-zero-to-skip}, \text{next non-zero value}\}$ 。我们用如下地址对应方法, 游长编码将更有效, 可能得到一长串 0: 用 Z 字扫描将 8×8 的矩阵  $\hat{F}(u, v)$  变为一个 64 个元素的向量, 如图 9-4 所示。大部分的图像块都有较小的空间高频分量, 量化后都变成了 0。因此 Z 字扫描很有可能碰到一长串 0。例如, 在图 9-2 中,  $\hat{F}(u, v)$  被转换为:

$$(32, 6, -1, -1, 0, -1, 0, 0, 0, -1, 0, 0, 1, 0, 0, \dots, 0)$$

中间有 3 个 0 串, 结尾处有一个长度为 51 的 0 串。

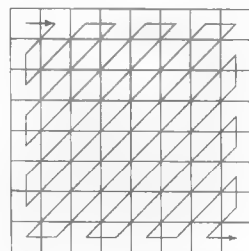


图 9-4 JPEG 中的 Z 形扫描

在 RLC 步骤里,  $\hat{F}$  的 AC 系数中的每个 0 串用 {RUNLENGTH, VALUE} 的数对方式表示值, 其中 RUNLENGTH 表示串里的 0 的数目, VALUE 表示下一个非 0 系数。为了尽量节省位数, 一个特殊的数字对 (0, 0) 紧跟着最后一个非 0 的 AC 系数, 表示到达了“块”的结尾。在上面提到的例子里, 不考虑前几个 (DC) 分量, 我们将得到:

(0, 6)(0, -1)(0, -1)(1, -1)(3, -1)(2, 1)(0, 0)

### 5. DC 系数的 DPCM 编码

DC 系数和 AC 系数分开编码。每一个 8×8 的图像块只有一个 DC 系数。DC 系数表示每个图像块的平均亮度, 所以不同的块 DC 系数会很大, 并且不同。但正如特征 1 所提到的, 在临近的块之间, DC 系数不会产生特别大的变化。这就使得可以采用理想的 DPCM 编码方式对 DC 系数进行编码。

如果前 5 个图像块的 DC 系数是 150、155、149、152、144, 那么 DPCM 编码后得到 150、5、-6、3、-8 (假设第  $i$  个块的预测器为  $d_i = DC_{i+1} - DC_i$ , 并且  $d_0 = DC_0$ )。我们期望 DPCM 编码后的幅值较小且变化不是很大, 这将为后面的熵编码步骤带来很大便利。

值得注意的是, AC 系数的游长编码需要对每一个块单独进行, 而 JPEG 中的 DC 系数的 DPCM 编码对整个图像实施一次即可完成。

260

### 6. 熵编码

DC 和 AC 系数最后都将进行熵编码。下面, 我们将讨论基本的熵编码 (即 baseline<sup>①</sup>), 该方法采用赫夫曼编码, 而且支持 8 位像素原图 (或彩色图像分量)。

下面我们将介绍两种熵编码方案, 对 DC 系数使用赫夫曼编码的变体, AC 系数的编码方案稍有不同。

#### (1) DC 系数的赫夫曼编码

每一个 DC 系数经过 DPCM 编码后可以用 (SIZE, AMPLITUDE) 表示, SIZE 表示需要多少位来表示 DC 系数, AMPLITUDE 表示实际使用的位数。

表 9-3 表示了和不同 AMPLITUDE 对应的 SIZE。注意, 表示 DPCM 的值需要超过 8 位, 而且 DPCM 的值有可能为负。我们用一个编码的取反来表示负数, 例如, 二进制编码 10 表示 2, 01 表示 -2; 11 表示 3, 00 表示 -3, 依此类推。在我们所提到的例子中, 码字 150、5、-6、3、-8 将会被表示为

(8, 10010110), (3, 101), (3, 001), (2, 11), (4, 0111)

表 9-3 基本熵编码细节——大小分类

大	小	幅 度
1		-1, 1
2		3, -2, 2, 3
3		-7, -4, 4, 7
4		-15, -8, 8, 15
⋮		⋮
10		-1023, ..., -512, 512, ..., 1023

在 JPEG 中, 对 SIZE 进行赫夫曼编码, 成为长度可变的编码。也就是说, 如果 SIZE2 出现过于频繁, 我们就有可能用一个位 (0 或 1) 来表示 SIZE 2。一般来说, 较小的 SIZE 值出现的次数比较频繁, 因为它们的熵较低, 所以, 赫夫曼编码可以带来压缩的效果。编码后, 定制的赫夫曼编码表可以保存在 JPEG 图像的头部, 否则要使用默认的赫夫曼编码表。

另外, AMPLITUDE 不进行赫夫曼编码。因为它的值变化范围较大, 进行赫夫曼编码得不到好的效果。

261

#### (2) AC 系数的赫夫曼编码

上面我们提到, AC 系数采用游长编码, 并且表示为一对数字 (RUNLENGTH, VALUE)。但

① JPEG 编码允许赫夫曼编码和算术编码, 二者都是熵编码方法。它还支持 8 位和 12 位的像素长度。

是, 在实际的 JPEG 实现中, 和 DC 系数一样, VALUE 用 SIZE 和 AMPLITUDE 的方式表示。为了节省位数, RUNLENGTH 和 SIZE 分别用 4 位来表示, 并且整合为一字节——我们称为 *Symbol 1*。*Symbol 2* 是 AMPLITUDE 值, 它的位数用 SIZE 表示:

**Symbol 1:** (RUNLENGTH, SIZE)

**Symbol 2:** (AMPLITUDE)

4 位的 RUNLENGTH 仅能表示长度为 0~15 的零串。有时候, 0 串的长度会超过 15, 这时候, 对 *Symbol 1* 使用特殊的扩展编码 (15, 0)。在最坏的情况下, 在 *Symbol 1* 结束前需要 3 个连续的 (15, 0), 这时候, RUNLENGTH 将是完全的实际游长。和 DC 编码类似, *Symbol 1* 采用赫夫曼编码, 而 *Symbol 2* 不采用。

### 9.1.2 JPEG 模式

JPEG 标准支持多种模式, 一些常用的模式有:

- 顺序模式
- 渐进模式
- 分级模式
- 无损模式

#### 1. 顺序模式

这是默认的 JPEG 模式。对灰度图或彩色图像分量进行从左到右、从上到下的扫描并编码。在我们的讨论中, 一直采用这种模式。Motion JPEG 视频就是对每一帧图像采用基本的顺序 JPEG 编码。

#### 2. 渐进模式

渐进式 JPEG 首先快速传送低质量的图像, 接着传送高质量的图像。这种模式在网页浏览中得到广泛使用。当网络带宽不是很高时, 这种多次扫描图像的方法非常有用。在渐进模式里, 前几次扫描只传递很少的位数和粗糙的图像轮廓。随着扫描次数的增加, 会接收到更多的数据, 图像画质也不断提高。这种做法的好处是用户可以通过前几次扫描接收到图像信息选择是否再继续接收数据以提高画面质量。

渐进式 JPEG 通过以下两种方式实现。主要的步骤 (DCT、量化等) 和顺序模式相同。

##### (1) 光谱选择

下列方案利用 DCT 系数的光谱 (空间频谱) 特性, 高等级的 AC 分量仅提供细节信息。

第 1 次扫描: 对 DC 和前几个 AC 分量编码, 例如 AC1、AC2。

第 2 次扫描: 对更多的 AC 分量编码, 例如 AC3、AC4、AC5。

:

第  $k$  次扫描: 对最后的几个 AC 分量编码, 例如 AC61、AC62、AC63。

##### (2) 连续近似

和渐进编码光谱带不同, 连续近似方法将所有的 DCT 系数同时编码, 最重要位 (Most Significant Bit, MSB) 最先编码。

第 1 次扫描: 对前几个 MSB 编码, 例如 Bits7、6、5 和 4。

第 2 次扫描: 对重要程度稍有降低的比特编码, 例如 Bit 3。

:

第  $m$  次扫描: 对最不重要的位 (LSB) 编码, 如 Bit0。

#### 3. 分级模式

顾名思义, 分级 JPEG 对处于不同分辨率层次中的图像进行编码。低分辨率的编码图像是通

过低通滤波器压缩后的图像,更高分辨率的图像提供更多的细节信息(和低分辨率图像提供的不同)。和渐进 JPEG 图像类似,分级 JPEG 可以通过多次扫描,渐进改善图像质量来传送。

图 9-5 给出了一个分为 3 级的 JPEG 编码器和解码器(两者用图中的虚线分隔)。

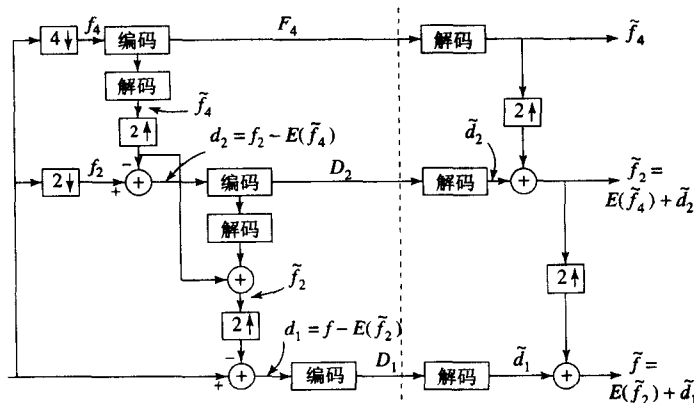


图 9-5 分级 JPEG 的块结构图

263

#### 算法 9-1 三级 JPEG 编码器

- 1) 降低图像分辨率。在输入图像  $f$  (例如  $512 \times 512$ ) 的每一维上除以 2 得到  $f_2$  ( $256 \times 256$ ), 从而降低输入图像分辨率。重复这一步骤得到  $f_4$  ( $128 \times 128$ )。
- 2) 压缩低分辨率图像  $f_4$ 。对  $f_4$  采用其他的 JPEG 方法(如顺序、渐进)编码, 得到  $F_4$ 。
- 3) 压缩差值图像  $d_2$ 。
  - (a) 对  $F_4$  解码得到  $\tilde{f}_4$ 。用某种插补方法扩展  $\tilde{f}_4$ , 得到和  $f_2$  同一分辨率的图像  $E(\tilde{f}_4)$ 。
  - (b) 对差值  $d_2 = f_2 - E(\tilde{f}_4)$  用某种 JPEG 方法(例如顺序或渐进)编码得到  $D_2$ 。
- 4) 压缩差值图像  $d_1$ 。
  - (a) 对  $D_2$  解码得到  $\tilde{d}_2$ ; 把它加到  $E(\tilde{f}_4)$  上得到  $\tilde{f}_2 = E(\tilde{f}_4) + \tilde{d}_2$ , 这是对  $f_2$  压缩又解压后得到的一个版本。
  - (b) 对差值  $d_1 = f - E(\tilde{f}_2)$  用某种 JPEG 方法(如顺序, 渐进)编码得到  $D_1$ 。

#### 算法 9-2 三层分级 JPEG 解码器

- 1) 对低分辨率图像  $F_4$  解压缩。使用和编码器相同的 JPEG 方法解压  $F_4$  得到  $\tilde{f}_4$ 。
- 2) 还原中间分辨率图像  $\tilde{f}_2$ 。应用  $E(\tilde{f}_4) + \tilde{d}_2$  得到  $\tilde{f}_2$ 。
- 3) 还原原始分辨率图像  $\tilde{f}$ 。应用  $E(\tilde{f}_2) + \tilde{d}_1$  得到  $\tilde{f}$ 。

需要指出的是,在编码器的步骤 3 中,差值  $d_2$  不是用公式  $f_2 - E(f_4)$  得到,而是用  $f_2 - E(\tilde{f}_4)$  得到。使用  $\tilde{f}_4$  是有代价的,因为在编码器里引入了一个额外的解码步骤,如图所示。

这确实有必要吗?有必要。因为解码器端不可能看到原始的  $f_4$ 。在解码器端的还原步骤里只可能使用  $\tilde{f}_4$  得到  $\tilde{f}_2 = E(\tilde{f}_4) + \tilde{d}_2$ 。因为压缩  $f_4$  使用有损的 JPEG 压缩方法时,  $\tilde{f}_4 \neq f_4$ , 所以编码器就必须在公式  $d_2 = f_2 - E(\tilde{f}_4)$  中采用  $\tilde{f}_4$  来避免在解码时产生不必要的错误。这种编码器/解码器步骤在很多压缩方案里使用。实际上,我们在 6.3.5 节里已经看到。之所以要这样做,是因为解码器只能访问被编码后的数据,不能访问原始数据。

同样,在编码器的步骤 4 里,  $d_1$  为  $f$  和  $E(\tilde{f}_2)$  的差值,而不是与  $E(f_2)$  的差值。

4. 无损模式

无损 JPEG 是 JPEG 的一种特殊情况，它没有图像质量的损失。正如在第 7 章里讨论的那样，它只采用了一种简单的微分编码方法，不涉及任何的变换编码。一般很少使用这种模式，因为它的压缩率远远低于其他的有损 JPEG 模式。但另一方面，它能满足一些特殊的需要，而且最新的 JPEG-LS 标准的目标就是进行无损的图像压缩（见 9.3 节）。

264

9.1.3 JPEG 位流概述

图 9-6 给出了 JPEG 图像的分层结构位流图。其中，帧（frame）就是一个图片，扫描（scan）就是对全部像素（例如，红色分量）的一次遍历，段就是一组块，块由 8×8 的像素组成。下面是一些头信息的例子：

- 帧头
  - 每像素的位数
  - 图像的宽和高
  - 分量数目
  - 唯一的 ID（对每一个分量而言）
  - 水平/垂直采样因子（对每一个分量而言）
  - 使用到的量化表（对每一个分量而言）
- 扫描头
  - 每个扫描中的分量数
  - 分量 ID（对每一个分量而言）
  - 赫夫曼/算术编码表（对每一个分量而言）

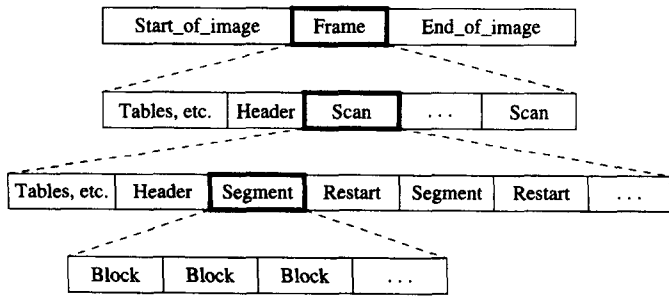


图 9-6 JPEG 位流

9.2 JPEG 2000 标准

毫无疑问，JPEG 标准是迄今为止最为成功和通用的图像格式。它成功的主要原因就在于它的在相对出色的压缩率下仍有很好的输出质量。然而，为了满足下一代图像应用的需求，JPEG 委员会定义了一个新的标准：JPEG 2000。

265

新的 JPEG 2000 标准[3]不仅在压缩率-失真间进行了更好的权衡，改善了图像的质量，而且新增了现有 JPEG 标准所缺乏的一些功能。特别是，JPEG 2000 标准将解决以下一些问题[4]：

- 低位率压缩 在中、高位率情况下，目前的 JPEG 标准在压缩率-失真方面有非常出色的表现。但是，如果码率低于 0.25bpp，图像失真就会变得无法接受。当我们使用一些有网络浏览功能的常用设备（如手表等）接收图像时，这一问题就很重要了。
- 无损和有损压缩 目前，还没有哪个标准能够在一个位流中提供很好的无损压缩和有损压缩。



- **大图像** 新的标准能处理分辨率超过 64K×64K 的图像而不会产生失真。它能处理的图像上限可达  $2^{32}-1$ 。
- **单一的解压体系结构** 现行的 JPEG 标准有 44 种模式, 其中的很多模式都是和某个应用有关的, 不能被大多数的 JPEG 解码器使用。
- **噪声环境中的传输** 新的标准改善了在有噪声环境中传输的差错恢复能力, 例如在无线网或因特网情况下。
- **渐进传输** 新的标准将会根据从低到高的码率提供无缝质量传输和有扩展性的解决方案。而且在压缩过程中, 目标码率和重组方案都不必考虑。
- **感兴趣区域编码** 新的标准允许指定感兴趣区域 (ROI), 该区域的编码质量优于图像的其他部分。例如, 我们希望对人脸的编码质量高一些, 而周围家具的编码质量可以略低。
- **计算机生成的影像** 现行的标准在处理自然图像时有很优秀的表现, 但处理计算机生成的影像时却差强人意。
- **复合文档** 新的标准提供了元数据的机制, 可以将额外的非图像数据整合到文件里。举个常见的例子, 当要同时包括图像和文本信息时, 这种方法很有用。

另外, JPEG 2000 能处理 256 路的信息, 而现行的 JPEG 标准只能处理 3 路颜色信息。卫星图像就能提供上述含有巨大数据量的信息。

因此, JPEG 2000 适用于各种应用。例如, 因特网、彩色传真、打印、扫描、数字图像、遥感、移动应用、医学图像、数字图书馆以及电子商务等。这种方法应用前景广阔, 而且可以远程浏览压缩过的图像。

JPEG 2000 标准有两种编码模式: 基于 DCT 和基于小波变换。提供基于 DCT 编码模式的目的是为了能够向后兼容现行的 JPEG 标准并实现基本 JPEG。所有新的功能和改进性能都是在基于小波变换模式下获得的。

266

### \*9.2.1 JPEG 2000 图像压缩的主要步骤

JEPG 2000 中使用的主要压缩方法是带有优化截断嵌入式块编码 (Embedded Block Coding with Optimized Truncation, EBCOT) 算法, 该算法最早是 Taubman[5]提出的。除了提高压缩的效率外, EBCOT 还产生了具有许多优秀特点的位流, 包括质量改进、分辨率伸缩和随机访问 (random access) 等。

EBCOT 的基本思想是首先将图像进行小波变换, 生成子带 LL、LH、HL、HH、再将这些子带划分成小块, 这些小块称为码块 (code block)。每一个码块都独立编码, 因而不会用到其他块的信息。

每一个码块产生独自的位流。使用基于块的编码方案, EBOCT 算法改进了差错恢复。EBOCT 算法包括以下三个步骤:

- 1) 块编码和位流的生成。
- 2) 压缩后比例失真 (PCRD) 优化。
- 3) 层格式化及表示。

#### 1. 块编码和位流的生成

由二维离散小波变换生成的子带首先被划分为  $32 \times 32$  或  $64 \times 64$  的小码块, 然后用 EBOCT 算法为每一个码块  $B_i$  生成伸缩率很高的位流。 $B_i$  对应的位流被独立地截为预先定义好的不同长度  $R_i^n$  的集合, 相应的失真度为  $D_i^n$ 。

对每一个码块  $B_i$  (见图 9-7), 设  $s_i[k] = s_i[k_1, k_2]$  为小码块的二维序列,  $k_1, k_2$  分别是行索引和列索引 (根据这个定义, 水平高通子带 HL 必须转置, 以使得  $k_1, k_2$  的含义和其他子带一致。这样转置意味着可以用同样的方式对待 HL 子带和 LH、HH、LL 子带, 同时可以使用相同的上下文模型)。

这个算法使用的无效区 (dead zone) 量化器如图 9-8 所示, 其中有两个单位长度的部分为 0。设  $\chi_i[k] \in \{-1, 1\}$  表示  $s_i[k]$  的符号, 同时设  $v_i[k]$  表示量化的幅值。我们有:

$$v_i[k] = \frac{\|s_i[k]\|}{\delta_{\beta_i}} \quad (9.2)$$

其中,  $\delta_{\beta_i}$  表示包含码块  $B_i$  的子带  $\beta_i$  的步长。设  $v_i^p[k]$  表示  $v_i[k]$  的二进制表示的第  $p$  位,  $p=0$  表示重要性最低的位,  $P_i^{\max}$  表示  $p$  的最大值, 这样若码块内至少有一个采样, 则  $v_i^{P_i^{\max}}[k] \neq 0$ 。

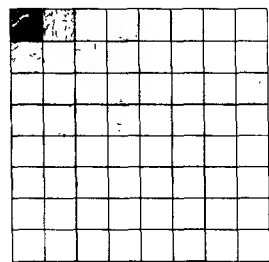


图 9-7 EBCOT 的码块结构

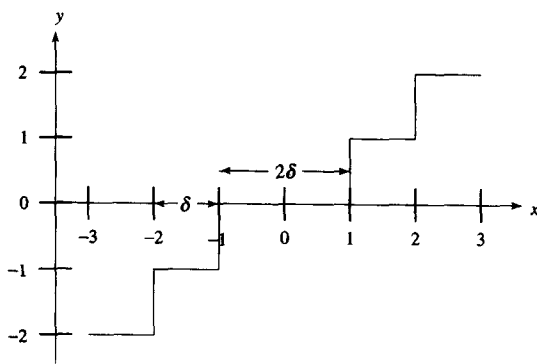


图 9-8 无效区量化器。无效量化区的长度是  $2\delta$ , 无效区的值量化为 0

编码过程很类似于一个位平面编码器, 最重要的位  $v_i^{P_i^{\max}}[k]$  最先编码, 接着是次重要的位  $v_i^{P_i^{\max}-1}[k]$ , 依次进行编码, 直到所有的位平面编码完毕为止。用这种方式, 如果位流被删减过, 那么码块中的一些采样将会损失一个或更多的不是很重要的位。这和使用一个比较粗糙的无效区量化器的效果相类似。

另外, 使用某个采样以前的编码信息和与之临近的采样的编码信息也很重要。在 EBCOT 中通过定义一个二值状态变量  $\sigma_i[k]$  来实现, 它的初始值为 0。当与之相关的采样的第一个非零位平面  $v_i^p[k]=1$  已经被编码, 则它的值由 0 变为 1。这个二值状态变量称为采样的“权” (significance)。

8.8 节介绍了一种数据结构——零树, 利用它可以很有效地编码小波系数的位流。零树数据结构的思想是把重要的采样聚簇在一起, 这样就使得我们有可能通过对一个二进制标记 (symbol) 编码实现对很多采样的处理。

EBCOT 利用了上面这个特性, 但是为了提高效率, 聚类思想仅用于相对较大的  $16 \times 16$  的子块。这样, 每一个码块又被划分为一系列二维的子块  $B_i[j]$ 。对每一个位平面, 明确的信息最先被编码, 以确认子块包含一个或多个重要采样。其他的子块则放到相应位平面的其余编码阶段中。

设  $\sigma^p(B_i[j])$  是位平面  $p$  中子块  $B_i[j]$  的“权”。我们使用四叉树对“权”图进行编码。通过确定带有叶子节点子块的来构造树, 即  $B_i^0[j] = B_i[j]$ 。高一层的节点用递归来定义:

$B_i^t[j] = \cup_{z \in \{0,1\}^2} B_i^{t-1}[2j+z], 0 \leq t \leq T$ 。树的根节点表示整个码块:  $B_i^T[0] = \cup_j B_i[j]$ 。

从根节点  $t=T$  开始, 每一次确定一层的码块的“权”, 直到叶节点  $t=0$  为止。“权”值被发送到算术编码器进行熵编码。如果“权”值有冗余则被跳过。如果发生下列情况, 则判定值是冗余的:

- 父节点是不重要的。
- 当前的四叉树已经在先前的位平面中认为是重要的。
- 是同一个重要的父节点下最后访问的四叉树, 并且其余的兄弟节点都是不重要的。

EBCOT 用 4 种不同的编码方法为位平面  $p$  内一个采样的新信息进行编码, 这 4 种方法如下:

- **零编码** 用来给  $v_i^p[k]$  编码, 假定量化采样满足  $v_i[k] < 2^{p+1}$ 。因为采样统计后发现可近似为马尔可夫链, 当前采样的“权”依赖于与之相近的 8 个直接邻居的值。这些邻居的“权”值可以分为以下三类:
  - **水平的**  $h_i[k] = \sum_{z \in \{1,-1\}} \sigma_i[k_1+z, k_2]$ , 其中  $0 \leq h_i[k] \leq 2$ 。
  - **垂直的**  $v_i[k] = \sum_{z \in \{1,-1\}} \sigma_i[k_1, k_2+z]$ , 其中  $0 \leq v_i[k] \leq 2$ 。
  - **对角线的**  $d_i[k] = \sum_{z_1, z_2 \in \{1,-1\}} \sigma_i[k_1+z_1, k_2+z_2]$ , 其中  $0 \leq d_i[k] \leq 4$ 。

码块外部的邻居被认为是不重要的, 但要注意子块不是完全独立的。256 种可能的邻居配置被简化为 9 种不同的上下文分配, 如表 9-4 所示。

- **游长编码** 游长编码的最初目的是产生串的 1 位权值的序列, 作为算术编码的准备。当一个有不重要邻居的不重要采样的水平序列出现时, 就采用这种编码方法而不是零编码。必须有以下四种之一情况发生时, 我们使用游长编码:
  - 连续四个采样都是不重要的。
  - 采样必须有不重要的邻居。
  - 采样都必须在一个子块内。
  - 第一个采样的水平索引  $k_1$  必须是偶数。

269

表 9-4 零编码的上下文分配

标号	LL、LH 和 HL 子带			HH 子带	
	$h_i[k]$	$v_i[k]$	$d_i[k]$	$d_i[k]$	$h_i[k]+v_i[k]$
0	0	0	0	0	0
1	0	0	1	0	1
2	0	0	>1	0	>1
3	0	1	x	1	0
4	0	2	x	1	1
5	1	0	0	1	>1
6	1	0	>0	2	0
7	1	>0	x	2	>0
8	2	x	x	>2	x

最后两种情况仅仅是为了提高效率。当有 4 个标记满足这些条件时, 我们将对一个特殊的位编码, 来确认在当前的位平面内是否有重要的采样 (使用一个单独的上下文模型)。如果四个采样里有任意一个是重要的, 那么第一个采样的索引将会用 2 位来表示。

- **符号 (sign) 编码** 对每一个采样至多执行一次符号编码, 当采样在零编码或游长编码过

程中发生从不重要到重要的转变时，立刻执行符号编码。因为每一个采样有 4 个水平邻居和 4 个垂直邻居，每个邻居都有可能是不重要的、正的或负的，一共有  $3^4=81$  种不同的配置。但是，利用水平和垂直的对称性，并且假设条件分布为  $\chi_i[k]$ ，对给定的任何邻居配置，与  $-\chi_i[k]$  的处理相同，最后上下文的数量减少到 5 种。

如果两个水平邻居都是不重要的，则  $\bar{h}_i[k]$  为 0；如果至少有一个水平邻居为正，则该值为 1；若至少有一个水平邻居为负，则该值为 -1（ $\bar{v}_i[k]$  的定义类似）。 $\hat{\chi}_i[k]$  表示符号的预测值，采用相关上下文编码的二进制标记是  $\chi_i[k] \cdot \hat{\chi}_i[k]$ 。表 9-5 给出了这些上下文的分配。

表 9-5   符号编码的上下文分配

标号	$\hat{\chi}_i[k]$	$\bar{h}_i[k]$	$\bar{v}_i[k]$
4	1	1	1
3	1	0	1
2	1	-1	1
1	-1	1	0
0	1	0	0
1	1	-1	0
2	-1	1	-1
3	-1	0	-1
4	-1	-1	-1

- **幅值细化**   这个过程的目的在于，给定  $v_i[k] \geq 2^{p+1}$  时，对  $v_i^p[k]$  的值编码。在这里仅用到 3 种上下文模型。第二个状态变量  $\sigma_i[k]$  在这里引入，当幅度细化第一次应用到  $s_i[k]$  时， $\sigma_i[k]$  的值从 0 变为 1。上下文模型和这个状态变量的值有密切的关系：当  $\sigma_i[k]=h_i[k]=v_i[k]=0$ ， $v_i^p[k]$  以上下文为 0 的方式编码，当  $\sigma_i[k]=0$  并且  $h_i[k]+v_i[k] \neq 0$  时，以上下文为 1 的方式编码， $\sigma_i[k]=1$  时以上下文为 2 的方式编码。

为了确保每个码块有一个合适的嵌入式位流，每一个位平面  $p$  的编码过程经过 4 个不同的通道，（ $\mathcal{P}_1^p$ ）到（ $\mathcal{P}_4^p$ ）：

- **前向重要性传播通道（ $\mathcal{P}_1^p$ ）**   子块的采样以行扫描线的顺序访问。不重要的采样和不满足邻居需求的采样将被跳过。对于 LH、HL 和 LL 子带，邻居的需求是至少有一个水平邻居是重要的。对于 HH 子带，邻居的需求是在四个对角线邻居中至少有一个是重要的。

对于满足了邻居需求的重要采样，零编码和游长编码将被采用来判断采样是否是第一次在位平面  $p$  内变得重要。如果判断为真，将通过符号编码对符号进行编码。这称为前向重要性传播通道，因为刚发现的重要性采样将在下面的新的权确定步骤里提供重要信息，引导扫描的方向。

- **反向重要性传播通道（ $\mathcal{P}_2^p$ ）**   这个过程和（ $\mathcal{P}_1^p$ ）相同，只是它是逆向的。邻居的需求有所放松，包括了在任意方向上至少有一个重要邻居的采样。
- **幅值细化通道（ $\mathcal{P}_3^p$ ）**   这一个通道处理那些是重要的但在前面两个通道中没有处理的采样。这些采样在幅值细化过程里进行处理。
- **规范化通道（ $\mathcal{P}_4^p$ ）**   用符号编码和游长编码对在前面 3 个编码过程中没有考虑的所有采样的  $v_i^p[k]$  编码。如果某个采样是重要的，则对它的符号立刻进行符号编码。

图 9-9 显示了每一个块的嵌入式位流的编码通道和四叉树编码的基本构成。 $S^p$  表示四叉树编码，标识位平面  $p$  内重要的子块。请注意，对任意一个位平面  $p$ ， $S^p$  只出现在最后的编码通道  $\mathcal{P}_4^p$

前,而不是在最初的编码通道  $\mathcal{P}^p$  前。这表明第一次在位平面  $p$  内变为重要的子块一直被忽略,直到最后一个编码通道才得到重视。

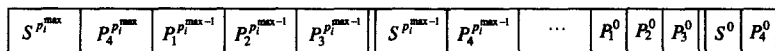


图 9-9 每个块的嵌入位流中编码扫描和四叉树代码

## 2. 压缩后率失真优化

在所有的子带采样被压缩后,就要进行压缩后率失真(Post Compression Rate Distortion, PCRD)步骤。PCRD 的目的是产生每一个码块的独立位流的优化截断,以便在位速率的约束条件下失真最小。每一个码块  $B_i$  的截断嵌入位流有速率  $R_i^{n_i}$ , 则重组图片后的失真如下所示(假设失真是可加的):

$$D = \sum_i D_i^{n_i} \quad (9.3)$$

$D_i^{n_i}$  表示从  $B_i$  得出的失真,  $B_i$  有截断点  $n_i$ 。对每一个码块  $B_i$ , 失真可以用下式计算得到:

$$D_i^{n_i} = w_{b_i}^2 \sum_{k \in B_i} (\hat{s}_i^n[k] - s_i[k])^2 \quad (9.4)$$

$s_i[k]$  是码块  $B_i$  内的子带采样的二维序列,  $\hat{s}_i^n[k]$  是和截断点  $n$  相关的采样的量化表示。值  $w_{b_i}^2$  是包含码块  $B_i$  的子带  $b_i$  小波基函数的  $L_2$  范数。

截断点  $n_i$  的优化选择问题可以转化为有以下约束条件的求最小值问题:

$$R = \sum_i R_i^{n_i} \leq R^{\max} \quad (9.5)$$

$R^{\max}$  表示可用的位速率,对于某一个  $\lambda$ , 任何使得表达式

$$(D(\lambda) + \lambda R(\lambda)) = \sum_i (D_i^{n_i^\lambda} + \lambda R_i^{n_i^\lambda}) \quad (9.6)$$

有最小值的截断点集合  $\{n_i^\lambda\}$  就是最优的。所以,找到在条件  $R(\lambda) = R^{\max}$  下使式(9.6)有最小值的截断点集合就能产生整个优化问题的解。

因为截断点的集合是离散的,所以不太可能找到一个  $\lambda$  使得  $R(\lambda)$  刚好等于  $R^{\max}$ 。但是,因为 EBCOT 算法使用相对较小的码块,每一个码块都有很多的截断点,所以存在足够多的机会找到最小的  $\lambda$ , 使得  $R(\lambda) \leq R^{\max}$ 。

很容易看出,每个码块  $B_i$  能单独地进行最小化。设  $N_i$  表示可行的截断点集合,并且令  $j_1 < j_2 < \dots$  是这些可行的截断点,它们的失真率比例由下面的表达式给出:

$$S_i^{j_k} = \frac{\Delta D_i^{j_k}}{\Delta R_i^{j_k}} \quad (9.7)$$

其中  $\Delta R_i^{j_k} = R_i^{j_k} - R_i^{j_{k-1}}$ ,  $\Delta D_i^{j_k} = D_i^{j_k} - D_i^{j_{k-1}}$ 。可以证明,斜率是严格递减的,因为失真率曲线是凸函数而且严格递减。 $\lambda$  值的最小化问题就简化为选择一个合适的  $\lambda$  使得

$$n_i^\lambda = \max\{j_k \in \mathcal{N}_i \mid S_i^{j_k} > \lambda\} \quad (9.8)$$

用两分法对失真率曲线进行操作就可找到最优值  $\lambda^*$ 。该方法的详细描述可以在[6]中找到。

## 3. 层的格式化和表示

与其他著名的可扩展图像压缩算法(如 EZW 和 SPIHT)只具有服务质量可扩展性不同,

EBCOT 算法同时提供了分辨率和服务质量的可扩展性。该功能通过分层的位流结构和双层的编码策略得到的。

EBCOT 产生的最终位流是由一组服务质量层组成的。服务质量层  $Q_l$  包含每个码块  $B_i$  初始的  $R_i^{n_l}$  个字节, 其他层  $Q_q$  包含了从码块  $B_i$  增加的  $L_i^q = R_i^{n_q} - R_i^{n_{q-1}} \geq 0$  个字节。其中  $n_q$  是与第  $q$  质量层的率失真域值  $\lambda_q$  有关的截断点。图 9-10 描述了分层的位流 (参见[5])。

273

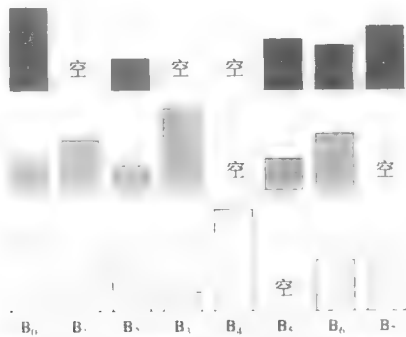


图 9-10 每层有 8 个块的 3 个质量层

除了增加的部分, 长度  $L_i^q$ 、新编码通道的个数  $N_i^q = n_i^q - n_i^{q-1}$ 、 $B_i$  对服务质量层  $Q_q$  产生第一次非空贡献的值  $p_i^{\max}$  以及  $B_i$  产生第一次非空贡献服务质量层的索引  $q_i$ , 这些辅助信息也需要被明确存储。这些辅助信息压缩在第二层编码引擎中。因此, 在这个双层体系结构中, 第一层产生了一个嵌入的块位流, 而第二层将块的贡献编码到每个质量层。

本节的重点是每个质量层的辅助信息的第二层处理。第二层编码引擎仔细处理两个显示模块间的冗余量。这两个量是  $p_i^{\max}$  和  $B_i$  产生第一次非空贡献的服务质量层的索引  $q_i$ 。

$q_i$  的编码是在每个子带中采用一个独立的嵌入式四叉树编码来实现的。在这个代表整个子带的树中, 设  $B_i^0 = B_i$  为叶子,  $B_i^T$  为树的根。令  $q_i' = \min\{q_j \mid B_j \subset B_i'\}$  为第一个四叉树  $B_i'$  中的任意编码模块有非空贡献的层的索引。用一个位来确定每一层  $t$  的每一个四叉树是否符合  $q_i' > q$ , 即有没有冗余的四叉树。如果  $q_i' < q-1$  或者某个父四叉树  $B_j^{t+1}$  的  $q_j^{t+1} > q$ , 则四叉树是冗余的。

另一个需要考虑的冗余量是  $p_i^{\max}$ 。很明显,  $p_i^{\max}$  只和服务质量层  $Q_q$  的编码相关。所以,  $p_i^{\max}$  中的多余信息不用被发送, 除非我们准备对  $Q_q$  进行编码。EBCOT 采用的是叶子驱动 (而不是树根驱动) 的经修改的嵌入式四叉树来完成上述工作。

令  $B_i'$  是每个子带的编码模块  $B_i$  顶部的四叉树的元素, 令  $p_i^{\max,t} = \max\{p_j^{\max} \mid B_j \subset B_i'\}$ 。另外, 令  $B_i'$  是  $B_i$  的祖先, 而  $P$  是一个保证大于任意编码模块  $B_i$  的  $p_i^{\max}$  的值。当编码模块  $B_i$  首次向服务质量层  $Q_q$  贡献位流时,  $p_i^{\max} = p_{i_0}^{\max,0}$  的值是用下面的算法编码的:

274

- 对于  $p = P-1, P-2, \dots, 0$ :
  - 发送二进制数位来确定  $p_i^{\max,t} < p$  是否成立跳过冗余位。
  - 如果  $p_i^{\max} = p$ , 则停止。

冗余位与条件  $p_i^{\max,t} < p$  相对应, 这个条件可以由满足  $p_{i_{t+1}}^{\max,t+1} < p$  的祖先或用来为不同编码模块  $B_j$  确定  $p_j^{\max}$  的局部四叉树编码推导得出。

### 9.2.2 使 EBCOT 适合 JPEG 2000

JPEG 2000 采用 EBCOT 算法作为主要的编码方法。但是, 这个算法作了少量的修改以提高

压缩率并减少计算复杂度。

为了进一步提高压缩效率，与原来在所有上下文中使用等概率状态来初始化熵编码器不同，JPEG2000 标准假设对于某些上下文分布很不对称，以此来减少对典型图像的模型适应代价。同时，对原有的算法进行了一些调整以进一步减少执行时间。

首先，用一个没有乘除法的低复杂度算术编码器（称为 MQ 编码器）代替原算法中的一般算术编码器。而且，JPEG 2000 不会移动 HL 子带编码模块，而是移动零编码上下文分配图的相应项。

为了保证扫描方向的一致性，JPEG 2000 将前向和后向传播通道结合为一个有邻近需求（相当于原来的后向通道）的向前重要性传播通道。而且，将子块的大小从  $16 \times 16$  减小到  $4 \times 4$  消除了对子块编码的需求。这样得到的小子块的概率分布是很不对称的，所以编码器的做法似乎所有的子块都是重要的。

这些修改的共同作用就是使软件执行的速度提高了 40%，而与原算法相比，平均丢失率大约是 0.15dB。

### 9.2.3 感兴趣区域编码

新的 JPEG 2000 标准的一个重要特性就是可以实现感兴趣区域（ROI）编码[8]。这样，相对于图像的背景或其他部分来说，某些部分可以采取高质量的编码。一个称为 MAXSHIFT 的方法，是一种可伸缩的方法，可以将 ROI 内的系数增加到更高层的位平面中。在这种嵌入式的编码过程中，所处理的比特将被放置到图像的非 ROI 部分前面。所以，给定一个缩小了的码率，ROI 将在图像的其他部分之前解码和改善。由于以上这些机制的作用，ROI 将会有着比背景图案更高的质量。

需要注意的一点是，在不考虑尺度因素时，位流的全解码能够以最大的逼真度重建整幅图像。图 9-11 显示了当样例图像的目标码率增加时，进行 ROI 编码的效果。

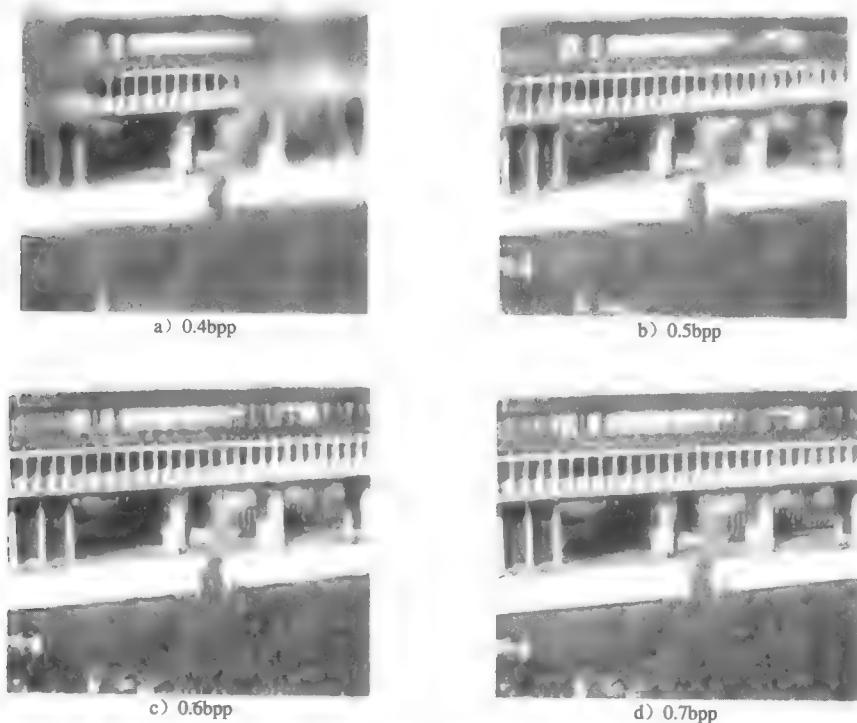


图 9-11 一幅图像的 ROI 编码，它使用圆形 ROI，具有渐增码率

275

276

### 9.2.4 JPEG 和 JPEG 2000 的性能比较

在研究过 JPEG 2000 的压缩算法的原理后,很自然会产生一个问题: JPEG 2000 的性能将在多大程度上优于以 JPEG 为代表的其他常用的标准?此前,已经有很多人对 JPEG 和其他标准进行过比较,这里我们将只比较 JPEG 2000 和 JPEG。

不同的判断标准(例如计算复杂度、压缩率和错误恢复等)均曾经用来评价一个系统的性能。由于我们主要关注 JPEG 2000 标准的压缩方面的性能,所以我们将只比较压缩率(对于其他判断标准感兴趣的读者可参考文献[9]和[10])。

给定一个固定的码率,我们将基于 PSNR 来定量比较压缩的图像的质量。对于彩色图像,通过计算所有 RGB 分量的均方差的平均值得到 PSNR。此外,我们将把通过 JPEG2000 和 JPEG 压缩的图像可视化地显示出来,从而作出定性的判断。我们对于 3 种类型的图像进行比较,即自然图像、计算机生成的图像,以及医学图像,每一类采用 3 幅图片。用于测试的图片可以在本书网站上的第 9 章的 Further Exploration 小节中找到。

对于每一幅图像,我们采用 JPEG 和 JPEG 2000 分别在 4 个码率下压缩,即 0.25bpp、0.5bpp、0.75bpp 和 1.0bpp。图 9-12 显示每一类图像的平均 PSNR 与码率的关系曲线。我们可以看出,在每一类中, JPEG 2000 的性能比 JPEG 都有很大程度的提高。

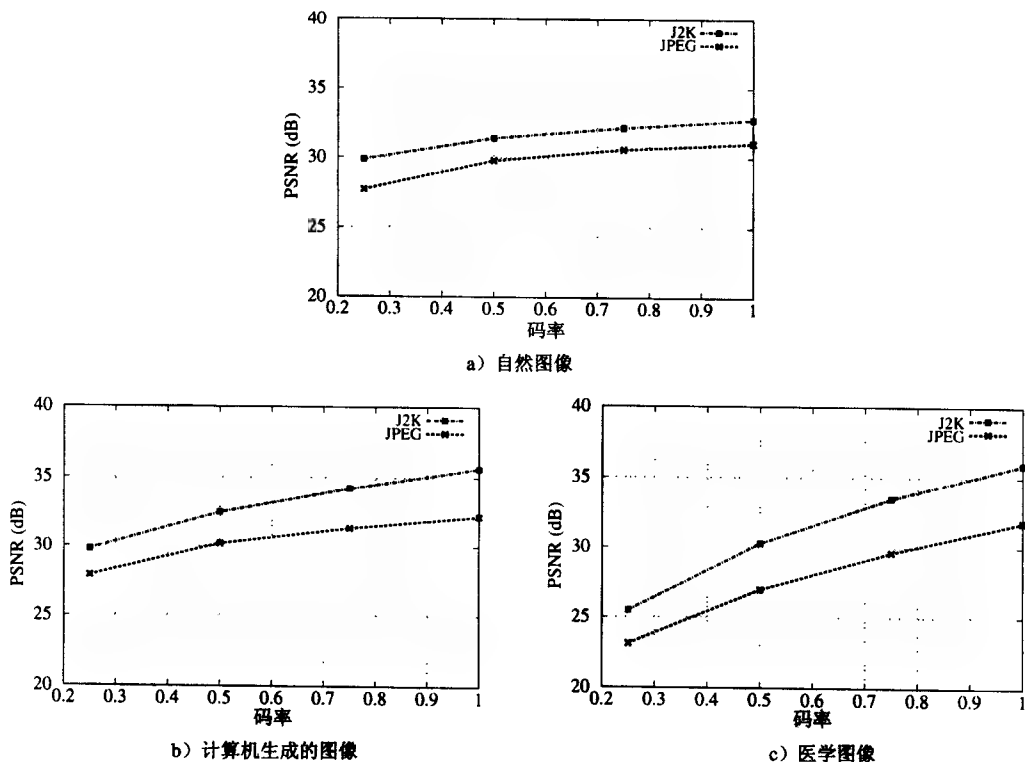


图 9-12 不同图像类型上的 JPEG 和 JPEG 2000 的性能比较

为了对于压缩结果有一个定性的认识,我们选择一幅图像,并显示应用两种算法在较低码率(0.75bpp)和较高码率(0.25bpp)下得到的解压缩后的效果。从图 9-13 的结果中可看出,采用 JPEG 2000 压缩的图像的处理痕迹较小。



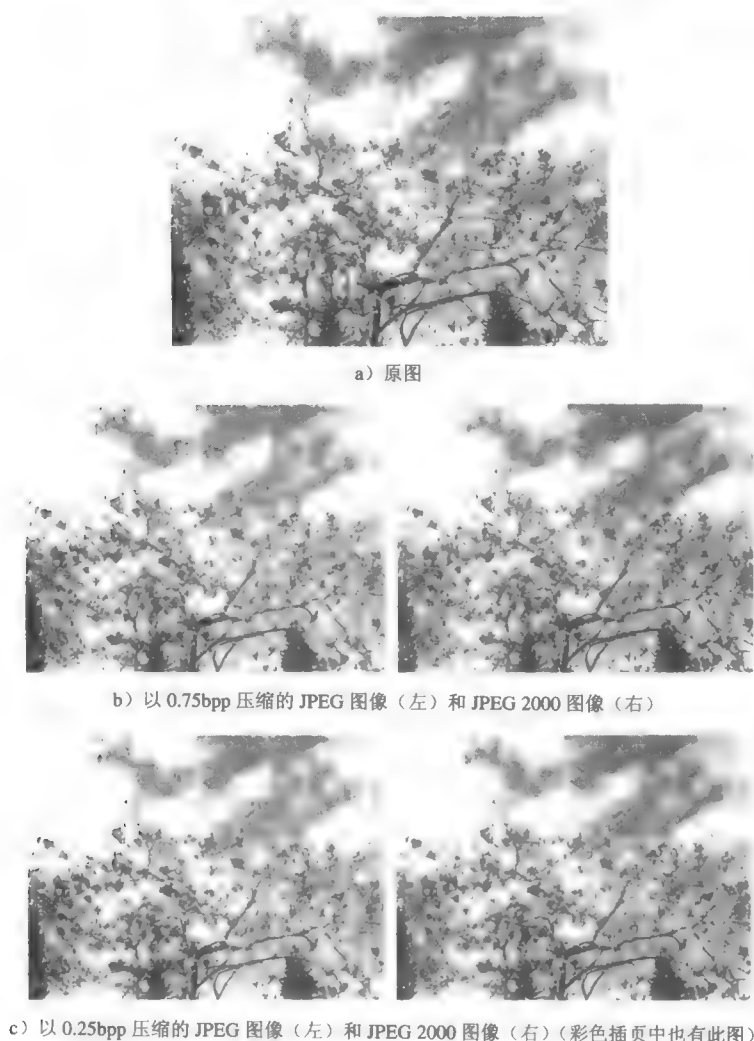


图 9-13 JPEG 和 JPEG 2000 的压缩

### 9.3 JPEG-LS 标准

通常来说,我们会采用一种无损压缩方案来处理某些重要的图像,如大脑的医学图像,或者较难取得的图像或者获取成本很高的图像。可以与 JPEG 2000 中的无损模式相媲美的标准是 JPEG-LS 标准,目的是实现无损编码[11]。JPEG-LS 与 JPEG 2000 相比,主要优点是采用的算法复杂度低。JPEG-LS 是 ISO 对医学图像建立更好标准的努力结果。

JPEG-LS 实际上是现有的 ISO/ITU 关于对连续色调的图像进行无损或“准无损”压缩的标准。JPEG-LS 的核心算法称为图像的低复杂度无损压缩算法(简记为 LOCO-I),是由惠普公司提出的[11]。该算法的设计基础是,降低复杂性通常要比采用更复杂的压缩算法使压缩结果稍有提高(指压缩的结果更小)更为重要。

LOCO-I 采用了上下文建模(context modeling)的概念。上下文建模的思想是利用输入源中的结构——在图像中每一个像素之后出现的像素值的条件概率。这种额外的信息叫做上下文。如果输入源中包含明显的结构是一种常见的情况,因此,我们可以使用比零阶熵更少的位数来实现压缩。

作为简单例子,我们假设有一个二进制源,它满足  $P(0)=0.4$  和  $P(1)=0.6$ ,那么第 0 阶熵  $H(S)=-0.4\log_2(0.4)-0.6\log_2(0.6)=0.97$ 。现在假设已经知道这个源有以下性质:如果前一个标记是 0,那么当前标记为 0 的概率为 0.8;而如果前一个标记是 1,那么当前标记为 0 的概率为 0.1。

如果我们用前一个标记为作为我们的上下文,我们就可以把输入标记看作两个集合,也就是上下文 0 和上下文 1。那么每一个集合的熵就分别是:

$$H(S_1)=-0.8\log_2(0.8)-0.2\log_2(0.2)=0.72$$

$$H(S_2)=-0.1\log_2(0.1)-0.9\log_2(0.9)=0.47$$

整个源的平均码率是  $0.4\times 0.72+0.6\times 0.47=0.57$ ,这远远小于本例中整个数据源的 0 次熵。

LOCO-I 实际上使用一种如图 9-14 所示的上下文模型。如果在光栅扫描顺序中,上下文像素  $a$ 、 $b$ 、 $c$  和  $d$  都会在当前像素  $x$  前显示,那么这就叫做因果上下文。

LOCO-I 可以分解为以下 3 个部分:

- 预测 用因果模板预测下一个样本  $x'$  的值。
- 确定上下文 决定  $x'$  出现的上下文条件。
- 残差编码 以  $x'$  的上下文为条件对预测的残差作熵编码。

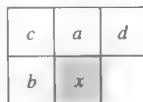


图 9-14 JPEG-LS 上下文模型

### 9.3.1 预测

一种更好的预测方法是使用基于计算局域边界方向的自适应模型。但是,因为 JPEG-LS 的目标是降低复杂度,LOCO-I 算法就采用了一种固定的预测模型以对检测到的垂直和水平边界作基本测试。这种算法使用的固定预测模型如下:

$$\hat{x}' = \begin{cases} \min(a,b) & c \geq \max(a,b) \\ \max(a,b) & c \leq \min(a,b) \\ a+b-c & \text{其他} \end{cases} \quad (9.9)$$

可以看出,这种预测器在三种简单的预测器间进行切换。它在当前位置左边有垂直边界的时候输出  $a$ ;在当前位置的上边有水平边界时输出  $b$ ;在临近样本都相对平滑的时候输出  $a+b-c$ 。

### 9.3.2 确定上下文

以当前预测误差(残差)为条件的上下文模型用由 3 个分量构成的上下文向量  $\mathbf{Q}=(q_1, q_2, q_3)$  进行索引,其中  $\mathbf{Q}$  的每一个分量分别为:

$$\begin{aligned} q_1 &= d-b \\ q_2 &= b-c \\ q_3 &= c-a \end{aligned} \quad (9.10)$$

这种差表示局部梯度,它记录的是局部平滑性或环绕当前样本的边界内容。因为差的取值范围比较广,而基本的上下文模型也是巨大的,从而使得直接上下文建模方法不可行。为解决问题,我们要使用参数缩减的方法。

一种有效的方法是把这些差量化,以便它们能用一些有限的值来表示。 $\mathbf{Q}$  的各分量可以用以下的量化器来定量化,该量化器以  $-T, \dots, -1, 0, 1, \dots, T$  为量化边界。在 JPEG-LS 中,  $T=4$ 。上下文的大小通过用  $-\mathbf{Q}$  取代第一个元素为负的上下文向量  $\mathbf{Q}$  得到进一步缩减。所以,不同上下文的状态数总共为

$\frac{(2T+1)^3+1}{2}=365$  种。然后,  $\mathbf{Q}$  就能被映射到  $[0, 364]$  的一个整数上了。

### 9.3.3 残差编码

对任意图像,预测残差都有一个有限的大小 $\alpha$ 。对一个预测值 $\hat{x}$ ,残差 $\varepsilon$ 则是在 $-\hat{x} \leq \varepsilon < \alpha - \hat{x}$ 范围内变动。因为 $\hat{x}$ 的值可以由解码器得到,残差 $\varepsilon$ 的动态范围可以通过对 $\alpha$ 取模,并且映射到 $-\left\lfloor \frac{\alpha}{2} \right\rfloor$ 和 $\left\lfloor \frac{\alpha}{2} \right\rfloor - 1$ 之间的一个值而减小。

可以证明,误差残差服从双面几何分布(Two-Sided Geometric Distribution, TSGD)。因此,它们可以基于 Golomb 码使用自适应的选择码进行编码,这对于服从几何分布的序列而言是最优的方法。

### 9.3.4 准无损模式

JPEG-LS 标准也提供准无损模式,其中,重建的样本与原来样本相差不超过某个 $\delta$ 。无损 JPEG-LS 模式可被看作误差模式 $\delta = 0$ 时的特例。准无损压缩可以使用量化得到:残差使用一个统一的以 $2\delta + 1$ 为间隔的量化器进行量化。量化后的 $\varepsilon$ 值用下式表示:

$$Q(\varepsilon) = \text{sign}(\varepsilon) \left\lfloor \frac{|\varepsilon| + \delta}{2\delta + 1} \right\rfloor \quad (9.11)$$

因为 $\delta$ 只能取很少几个整数值,所以这种除法运算可以使用查找表高效率地实现。在准无损模式中,前面描述的预测和上下文决定步骤都只能对通过量化后的值进行。

281

## 9.4 二值图像压缩标准

由于越来越多地使用电子形式来处理文档,因此越来越需要能够有效压缩二值图像(指的是仅用1位表示黑或白像素)的方法。常见的例子就是传真图像。利用图像数据的二元特性的算法的性能一般要好于通常的图像压缩算法。早期的传真标准(如G3和G4)使用简单的二值图像的结构模型。图像上的每一扫描线都被看作黑白像素组成的序列。然而,考虑到相邻像素和待编码的数据的特征就能够构造出更加有效的算法。本节将分析 JBIG 标准和它的后续标准 JBIG2,以及这两种标准的制订原因和原则。

### 9.4.1 JBIG 标准

JBIG 是由 Joint Bi-level Image Processing Group (联合二值图像专家组)提出的二值图像的编码标准。这种无损压缩标准主要用来为打印的图像或者手写的文本、由计算机产生的文字和传真进行编码。它具有渐进的编码和解码能力,也就是说,编码后的位流包含一套渐进的高分辨率的图像。这种标准也可以用来独立地为每一个位平面来编码灰度和彩色图像,但这并不是主要的目的。

JBIG 压缩标准具有3种独立的操作模式:渐进式、渐进-兼容序列式和单渐进序列式。渐进-兼容序列模式使用与渐进模式一致的位流,唯一的不同是,在这种模式下数据被分成“条”。

单渐进序列模式具有唯一的最低的分辨率层。因此,可以在不参照其他较高分辨率层的情况下为整幅图像编码。这两种模式都可以看成是渐进模式的特例。因此,我们的讨论仅仅涉及渐进模式。

JBIG 编码器可以分解成两个部分:

- 分辨率缩减和差分层编码器
- 最低分辨率层编码器

输入图像通过一系列分辨率缩减和差分层编码器。每一个编码器在功能方面是相同的,只是它们的输入图像具有不同的分辨率。一些 JBIG 标准的实现会选择迭代地使用同一个物理编码器。最低分辨率的图像使用最低分辨率层编码器来进行编码。此编码器的设计比分辨率缩减和差分层

编码器要简单一些,因为分辨率缩减和判决预测操作不是必需的。

### 9.4.2 JBIG2 标准

282

尽管 JBIG 标准提供无损和渐进的(从有损和无损)编码功能,此标准产生的有损图像与原始图像相比,质量上相差很多,因为无损图像的像素数目最多只能有原始图像像素数目的 1/4。相比而言,JBIG2 标准用于无损、无损和有损至无损图像的压缩。JBIG2 的目标不仅在于提供比已有标准更好的无损压缩性能,而且要能够融合有损压缩标准,在提高压缩率的前提下,尽可能减少视觉下降。

JBIG2 的独特之处在于它具有质量渐进和内容渐进。质量渐进的意思是,它的位流和 JBIG 标准的表现相似,在 JBIG 标准中,图像质量从低向高(甚至可能无损)渐进。另一方面,内容渐进允许不同类型的图像数据可以渐进相加。JBIG2 编码器将输入的二值图像分解成具有不同属性的区域,并且对每一部分使用不同的方法分别进行编码。

和其他的图像压缩标准相比,仅有 JBIG2 位流和解码器是明确定义的。因此,任何能够产生正确的位流的编码器都是“适应的”,而不论其实际采取的方式。另一个使 JBIG2 标准不同于其他编码标准的特征是,它可以表示一个文件中的一个文档的多个页面,可以使它利用页间的相似性。

例如,如果一个字符在一页中出现,它也可能在其他页中出现。使用一种基于字典的技术,这一字符只需编码一次,而不是在它出现的每一页上多次编码。这一压缩技术和视频编码有些类似,它可以利用帧间的冗余来提高压缩率。

JBIG2 可以进行内容渐进编码和通过基于模型的编码提供较好的压缩性能。在基于模型的编码中,在一个图像中为不同的数据构造不同的模型,这样就可实现附加的编码增益。

#### 1. 基于模型的编码

基于模型的编码的思想和基于上下文的编码基本相同。从后来的研究中,我们发现,可以通过仔细设计一个上下文模板并精确地估计每个上下文的概率分布来实现更好的压缩性能。同样,如果我们将图片内容分成不同的种类,并且针对每一类推导出一个模型,那么就很有可能为数据的行为建立精确的模型,进而得到比较高的压缩率。

在 JBIG 风格的编码中,自适应的和模型模板能够获取图像的结构。这种模型是一般化的,即它适用于各种类型的数据。但一般化也意味着它不能直接处理文本和网板数据之间的结构上的不同,而它们几乎构成了所有的二值图像的内容。JBIG2 为这些数据类型设计了定制化的模型,从而有效利用了这一特征。

JBIG2 规范要求编码器首先将输入图像分割成不同数据类型的区域,特别是文本和半色调区域,然后每个区域再根据各自的特征分别编码。

#### (1) 文本区域编码

283

每一个文本区域可以分成包含有联系的黑色像素的像素块。这些“块”对应于组成这一区域内容的字符。然后,为这个字符有代表性实例的位图进行编码并放入字典,而不是对这一字符的所有像素进行编码。对于被编码的任何字符,算法首先在字典中寻找与其匹配的字符。如果找到匹配字符,那么指针指向字典中相应的项,并且对页面上字符的位置编码。否则,像素块被直接编码,并加入到字典中。这一技术在 JBIG2 规范中称作模式匹配和置换。

然而,对于扫描文档来说,同一字符的两个实例不太可能每个像素都匹配。在这种情况下,JBIG2 使用包括精化数据的选项来重塑页面上的原始字符。精化数据使用字典中的匹配字符的像素为当前字符进行编码。编码器能自由选择精化数据,使之或是精确的或是有损的。这一方法叫做软模式匹配。

数值数据（如字典中匹配字符的索引和页面上字符的位置）可进行逐位编码或赫夫曼编码。字典中字符的每个位图通过基于 JBIG 的技术进行编码。

## (2) 半色调区域编码

JBIG2 方法推荐了两种针对半色调图像进行编码的方法。第一种方法类似于在 JBIG 中使用的基于上下文的算术编码。唯一的不同在于，新标准允许上下文模板包含多至 16 个模板像素，其中四个可以是自适应的。

第二种方法称为去网 (descreening)。此方法涉及转换为灰度图并对灰度值进行编码。在此种方法中，二值区域被分成大小为  $m_b \times n_b$  的块。对于  $m \times n$  二值区域来说，所得的灰度图像的维数是  $m_g = \lfloor (m + (m_b - 1)) / m_b \rfloor$  和  $n_g = \lfloor (n + (n_b - 1)) / n_b \rfloor$ 。然后计算灰度值，作为对应于  $m_b \times n_b$  块的二元像素值的和。灰度图像的位平面用基于上下文的算术编码方法编码。灰度值用作半色调位图模式的字典的索引。解码器使用这些值来查询字典，从而重建原来的半色调图像。

## 2. 预处理和后处理

JBIG2 允许使用有损压缩，但并没有指定具体方法。从解码器的角度来看，对于用编码器编码的解码图像的位流是无损的，虽然对于原始的图像并不一定是这样。编码器可以预先修改输入图像来提高编码效率。预处理器通常使用一种一般不影响图像外观的方法来修改原始图像，从而降低编码长度。一般来说，此操作去除噪声像素并使像素块更加平滑。

后处理（是规范中没有涉及的话题）对于半色调来讲是特别有用的，可能产生视觉效果更好的图像。将解码后的图像输出到某一个具体输出设备（像激光打印机等）有助于调整。

## 9.5 进一步探索

Pennebaker 和 Mitchell[1]及 Taubman 和 Marcellin[3]分别对 JPEG 和 JPEG2000 进行了详细介绍。Bhaskaran 和 Konstantinides[2]详细讨论了几种图像压缩标准及其理论。

284

本书网站的 Further Exploration 提供了一个用 Java 写成的 JPEG demo，可以用它来进行实验。其他有用的链接包括：

- 用来评估 JPEG/JPEG 2000 性能的测试图像库，其中包括自然图像、计算机生成的图像以及一些医学图像。
- 更多 JPEG 和 JPEG 2000 的相关链接。
- JPEG 2000 编/解码器的 Java 以及 C 实现。
- 用来比较 JPEG 和 JPEG 2000 的 Java Applet。
- 基于上下文的图像压缩的简单说明。
- 几篇研究 LOCO-I 的文章。
- JPEG-LS 的公开源代码。
- 对 JBIG 的介绍和文档，以及 JBIG 和 JBIG2 的源代码。
- Mark Nelson 编译的数据压缩的资源，包括库、文档和 JPEG、JPEG 2000、JPEG-LS 和 JBIG 的源代码等。

## 9.6 练习

1. (a) JPEG 使用离散余弦变换 (DCT) 进行图像压缩。

i. 图像  $f(i, j)$  如下所示，计算  $F(0,0)$  的值？

ii. 对于这个  $f(i, j)$  来说，最大的 AC 系数  $|F(u, v)|$  是什么？为什么？这时  $F(u, v)$  是正数

还是负数? 为什么?

20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
80	80	80	80	80	80	80	80
80	80	80	80	80	80	80	80
140	140	140	140	140	140	140	140
140	140	140	140	140	140	140	140
200	200	200	200	200	200	200	200
200	200	200	200	200	200	200	200

(b) 详细说明一个三层 JPEG 将如何对上面的图像编码, 假设:

- i. 三层中的所有编/解码器都使用无损 JPEG 压缩。
- ii. 平均将每个  $2 \times 2$  的块减少至一个像素值。
- iii. 扩充复制一个像素值四次。

285

2. 在 JPEG 中, 离散余弦变换作用于图像的  $8 \times 8$  块上, 为了区别新的算法, 我们称之为 DCT-8。一般而言, 我们能够定义一个 DCT- $N$  来对图像中一个  $N \times N$  的块进行 DCT 变换, DCT- $N$  定义如下:

$$F_N(u, v) = \frac{2C(u)C(v)}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \cos \frac{(2i+1)u\pi}{2N} \cos \frac{(2j+1)v\pi}{2N} f(i, j)$$

$$C(\xi) = \begin{cases} \frac{\sqrt{2}}{2}, & \xi = 0 \\ 1, & \text{其他情况} \end{cases}$$

给定  $f(i, j)$ , 给出你得到的  $F_2(u, v)$  (即给出对以下图像做 DCT-2 变换的结果)。

100	-100	100	-100	100	-100	100	-100
100	-100	100	-100	100	-100	100	-100
100	-100	100	-100	100	-100	100	-100
100	-100	100	-100	100	-100	100	-100
100	-100	100	-100	100	-100	100	-100
100	-100	100	-100	100	-100	100	-100
100	-100	100	-100	100	-100	100	-100
100	-100	100	-100	100	-100	100	-100

3. 根据上面定义的 DCT- $N$ ,  $F_N(1)$  和  $F_N(N-1)$  分别是表示最低和最高空间频率的 AC 系数。
- (a) 已知在图像滤波时,  $F_{16}(1)$  和  $F_8(1)$  不能捕获相同的 (最低) 频率响应, 解释原因。
  - (b)  $F_{16}(15)$  和  $F_8(7)$  能够捕获同样的 (最高) 频率响应吗?
4. 假设有一张电脑卡通图片和一幅照片, 如果你能够用 JPEG 或者 GIF 两种图像压缩方法对这两副图像进行压缩, 你会对这两种图像分别应用哪种方法? 证明你的结论。
5. 假设我们看到一幅解压缩的  $512 \times 512$  像素的 JPEG 图像, 但是这幅图像只有颜色部分的存储信息 (而没有亮度部分) 来进行解压缩。这幅  $512 \times 512$  像素的彩色图像看上去会怎么样? 假设 JPEG 使用 4:2:0 方案压缩。
6. (a) JPEG 有多少主要模式? 它们叫什么名字?
- (b) 在分级模型下, 简要解释在传送图像到解码端时为什么必须要在编码端有一个编码/解码

循环。

(c) 为了能够快速、粗粒度地显示图像并且逐步提高图像质量, 可以应用哪两种方法仅对 JPEG 文件的信息部分解码?

7. 我们能够在普通的 JPEG 图像中使用基于小波变换的压缩方法吗? 如何使用?
8. 为了能让外来物种看到图像, 我们决定创造一种基于 JPEG 的新的图像压缩标准。JPEG 工作流程中的哪一部分是需要改变的?
9. 与 EZW 不同, EBCOT 不会显式利用小波系数的空间关系。它使用的是 PCRD 优化方法, 讨论这种方法的合理性。
10. JPEG 2000 数据流的信噪比 (SNR) 是否可调? 如果是, 解释如何使用 EBCOT 算法实现调整。
11. 实现编码器和解码器的 3 层 JPEG 算法的代码转换、量化和分层编码。你的代码必须 (至少) 包含一个显示结果的图像用户界面。你不必实现熵 (无损) 编码, 你可以选择性地包含一些公开的源代码。

286

## 9.7 参考文献

- [1] W.B. Pennebaker and J.L. Mitchell, *The JPEG Still Image Data Compression Standard*, New York: Van Nostrand Reinhold, 1993.
- [2] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed., Boston: Kluwer Academic Publishers, 1997.
- [3] D.S. Taubman and M.W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Norwell, MA: Kluwer Academic Publishers, 2002.
- [4] C. A. Christopoulos, "Tutorial on JPEG2000," In *Proc. of Int. Conf. on Image Processing*, 1999.
- [5] D. Taubman, "High Performance Scalable Image Compression with EBCOT," *IEEE Trans. Image Processing*, 9(7): 1158–1170, 2000.
- [6] K. Ramachandran and M. Vetterli, "Best Wavelet Packet Basis in a Rate-Distortion Sense," *IEEE Trans. Image Processing*, 2: 160–173, 1993.
- [7] I. Ueno, F. Ono, T. Yanagiya, T. Kimura, and M. Yoshida, *Proposal of the Arithmetic Coder for JPEG2000*, ISO/IEC JTC1/SC29/WG1 N1143, 1999.
- [8] A. N. Skodras, C. A. Christopoulos, and T. Ebrahimi, "JPEG2000: The Upcoming Still Image Compression Standard," In *11th Portuguese Conference on Pattern Recognition*, pp. 359–366, 2000.
- [9] D. Santa-Cruz and T. Ebrahimi, "A Study of JPEG2000 Still Image Coding Versus Other Standards," In *X European Signal Processing Conference*, pp. 673–676, 2000.
- [10] D. Santa-Cruz, T. Ebrahimi, J. Askelof, M. Larsson, and C. A. Christopoulos, *JPEG2000 Still Image Coding Versus Other Standards*, ISO/IEC JTC1/SC29/WG1 (ITU-T SG8), 2000.
- [11] M. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS," Technical Report HPL-98-193R1, Hewlett-Packard Technical Report HPL-98-193R1, 1998.
- [12] N. Merhav, G. Seroussi, and M. J. Weinberger, "Optimal Prefix Codes for Sources with Two-Sided Geometric Distributions," *IEEE Transactions on Information Theory*, 46(1): 121–135, 2000.

287

## 第 10 章 基本视频压缩技术

正如第 7 章中所讨论的,未压缩的视频数据量非常庞大。一个画面大小为  $352 \times 288$  的普通 CIF (公共媒介格式) 视频文件在没有压缩的情况下,将占用 35Mbps 的带宽。在 HDTV 中,码率就会很容易超过 1Gbps。这就对视频数据的存储以及网络的通信能力提出了新的问题和挑战。

本章介绍一些基本的视频压缩技术,并结合 H.261、H.263 这两个主要针对视频会议的视频压缩标准来进行具体的阐述。接下去的两章将进一步介绍几种 MPEG 的视频压缩标准,包括最新的 H.264 标准。

### 10.1 视频压缩简介

视频是由一系列的时间上有序的图像(我们叫做帧)所组成的。解决视频压缩的一个简单的方案就是基于前面的帧的预测编码。举个例子,假设我们构造一个预测器,预测器的预测结果和前一帧相同。压缩不是对图像本身进行相减,而是按照时间顺序进行相减,并将残差进行编码。

假设大多数的视频并不是随时间变化的,那么我们得到的柱状图在 0 值处有一个很陡的尖峰,也就是说,根据原视频的熵可以进行很大的压缩,这正是我们所期望的。

尽管如此,事实证明在我们可以接受的代价范围内,我们可以通过搜索图像中的适当部分并和前一帧相减来获得更好的压缩效果。毕竟,我们简单的相减方案在办公家具以及大学照片这样静止的背景下可能是十分有效的。但是在足球比赛中,画面上有很多快速运动的球员,当与静态的绿色球场相减的时候就会产生大量的数据。

所以,在下一节中,我们将探讨如何进行更好的视频压缩。在下一帧中寻找足球运动员的位置的策略叫做运动估计(motion estimation);来回移动帧的位置为了最大程度上将球员从图像中减去,叫做运动补偿(motion compensation)。

### 10.2 基于运动补偿的视频压缩

前面几章中所讨论的图像压缩技术(如 JPEG 和 JPEG2000)采用了空间冗余(spatial redundancy)。图像内容在整个图片上变化比较缓慢这个现象使得空间维度上高频分量的大量压缩成为可能。

一段视频可以看作在时间(temporal)维度上顺序播放的一系列图像。由于视频的帧率通常比较高(大于 15 帧每秒),并且摄像头的参数(焦距、位置、视角等)变化较慢,所以连续帧的图像内容是很相似的,除非有移动较快的物体。换句话说,视频在时间维度上存在冗余。

时间冗余通常比较显著,利用这个特征,不必将每帧视频图像都作为一幅新的图像进行编码,而是将当前帧和其他帧的差值进行编码。如果帧间的时间冗余足够大,那么不同的图像只含有少量的信息和比较低的熵,这对压缩来说是非常有利的。

前面提到过,一个最简单的生成差值图像的方法就是将一张图像按照像素点值减去另一张图像。但是用这种方法无法实现高压缩率。由于帧间图像的主要差别是由摄像头或者物体的运动造成的,所以可以通过在这些帧里探测相应像素或区域的移动并测量它们的差值来“补偿”这些运动生成器。采用该方法的视频压缩算法称为基于运动补偿(MC)的压缩算法。这些算法的三个主要步骤是:



- 1) 运动估计(运动向量查找)。
- 2) 基于运动补偿的预测。
- 3) 预测误差的生成——差值。

为提高效率,我们把每张图像分为大小为 $N \times N$ 的宏块(macroblock)。默认情况下,亮度图的 $N$ 值取16。对于色度图来说,如果采用4:2:0的采样格式,则 $N$ 值为8。运动补偿在像素级别和视频对象(video object)级别(如MPEG-4)并不起作用,而是在图像的宏块级别起作用。

当前帧称为目标帧(target frame)。我们要在目标帧中的宏块和参考帧(前向帧或后向帧)中最相似的宏块间寻找匹配。在这种情况下,目标宏块由参考宏块预测生成。

参考宏块到目标宏块的位移称作运动向量(MV)。图10-1描述了前向预测(forward prediction)的情况,前向预测用以前的帧作为参考帧进行预测。如果参考帧为以后的帧,则被称为后向预测(backward prediction)。这两个宏块间的差值就是预测误差。

对于基于运动补偿的视频压缩来说,在第一帧后,只需要对运动向量和差值宏块编码,因为这些信息足以用于解码并重新生成完整的图像。

在下一节中,我们将讨论运动向量的查找算法,接下来讨论一些常用的视频压缩标准,其中包括运动向量的搜索算法。

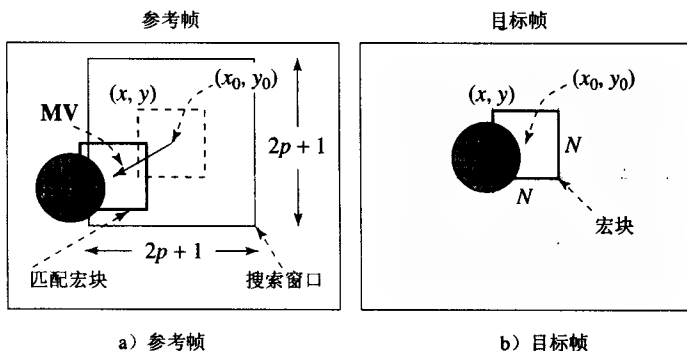


图10-1 视频压缩中的宏块和运动向量

### 10.3 搜索运动向量

前面定义的运动向量 $MV(u, v)$ 的搜索是一个匹配问题,也称为相关性(correspondence)问题[1]。由于运动向量搜索的计算十分复杂,所以常限制在一个较小的相邻区域内。水平位移 $i$ 和垂直位移 $j$ 必须在 $[-p, p]$ 的范围之内,其中 $p$ 是一个取值较小的正整数。图10-1所示的搜索窗口大小为 $(2p+1) \times (2p+1)$ 。宏块的中心 $(x_0, y_0)$ 可以放在窗口中的任何一个单元格中。

为了方便起见,我们用目标帧中左上角的坐标值 $(x, y)$ 作为宏块的原点。设 $C(x+k, y+l)$ 为目标(当前)帧的宏块中的像素,  $R(x+i+k, y+j+l)$ 为参考帧的宏块中的像素,其中 $k$ 和 $l$ 代表宏块中的像素的索引,  $i$ 和 $j$ 分别为水平和垂直的位移。两个宏块的差可以用它们的平均绝对误差(Mean Absolute Difference, MAD)来测量,定义为:

$$MAD(i, j) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} |C(x+k, y+l) - R(x+i+k, y+j+l)|, \quad (10.1)$$

其中 $N$ 为宏块的大小。

搜索的目标是找到一个向量 $(i, j)$ 作为运动向量 $MV = (u, v)$ , 使 $MAD(i, j)$ 取最小值:

$$(u, v) = [(i, j) | MAD(i, j) \text{ is minimum}, i \in [-p, p], j \in [-p, p]] \quad (10.2)$$

在前面的讨论中,我们使用了平均绝对误差。尽管如此,这绝不是唯一的可选的方案。实际上,一些编码器(比如H.263)会采用绝对误差和(Sum of Absolute Difference, SAD)的误差测量方法,还可以采用一些其他常用的误差测量方法,比如均方差(Mean Square Error, MSE)。

#### 10.3.1 顺序搜索

寻找运动向量最简单的方法是顺序搜索参考帧中整个 $(2p+1) \times (2p+1)$ 大小的窗口(也称为全

搜索)。将该窗口中的每一个宏块逐个像素地和目标帧中的宏块进行比较,从式(10.1)得到它们各自的 MAD。MAD 最小的向量  $(i, j)$  即为目标帧中宏块的运动向量  $MV(u, v)$ 。

程序 10-1 运动向量: 顺序搜索

---

```

BEGIN
  min_MAD = LARGE_NUMBER; /* Initialization */
  for i = -p to p
    for j = -p to p
      {
        cur_MAD = MAD(i, j);
        if cur_MAD < min_MAD
          {
            min_MAD = cur_MAD;
            u = i; /* Get the coordinates for MV. */
            v = j;
          }
      }
  }
END

```

---

显然,顺序搜索算法的代价是相当高的。从式(10.1)可以看出,每一个像素的比较需要三个操作(相减、绝对值、相加)。因此获取一个宏块的运动向量的复杂度为  $(2p+1) \cdot (2p+1) \cdot N^2 \cdot 3 \Rightarrow O(p^2 N^2)$ 。

举一个例子,假设视频的分辨率为  $720 \times 480$ , 帧率为 30fps。再假设  $p=15$ ,  $N=16$ , 那么搜索每一个运动向量的运算量为:

$$(2p+1)^2 \cdot N^2 \cdot 3 = 31^2 \times 16^2 \times 3.$$

考虑到一个图像帧有  $\frac{720 \times 480}{N \cdot N}$  个宏块,每秒有 30 帧图像,所以每秒的运算量为:

$$\begin{aligned}
 \text{OPS\_per\_second} &= (2p+1)^2 \cdot N^2 \cdot 3 \cdot \frac{720 \times 480}{N \cdot N} \cdot 30 \\
 &= 31^2 \times 16^2 \times 3 \times \frac{720 \times 480}{16 \times 16} \times 30 \approx 29.89 \times 10^9.
 \end{aligned}$$

这显然使得视频的实时编码变得十分困难。

### 10.3.2 2D 对数搜索

对数搜索(Logarithmic Search)虽然不是最优方法,但通常是非常有效的一个办法,而且代价较低。2D 对数搜索的方法搜索运动向量的过程中需要进行多次迭代,类似折半查找过程。如图 10-2 所示,在搜索窗口中只有 9 个位置被标记为“1”,它们作为基于平均绝对误差搜索的起始位置。当 MAD 最小值的位置确定后,将新的搜索区域中心移动到该位置,搜索的步长(偏移)减半。在下次迭代中,9 个新的位置被标记为“2”,依此类推<sup>①</sup>。设目标帧中宏块的中心位置为  $(x_0, y_0)$ , 搜索过程如下:

程序 10-2 运动向量的 2D 对数搜索

---

```

BEGIN
  offset = [  $\frac{p}{2}$  ];
  Specify nine macroblocks within the search window in the Reference frame,
  they are centered at  $(x_0, y_0)$  and separated by offset horizontally and/or vertically;
  WHILE last  $\neq$  TRUE

```

---

① 这个过程是启发式的。它假设图像内容具有一般意义上的连贯性(单调性)——在搜索窗口中的图像不会随机变化。否则,这个过程就很难找到最好的匹配。

```

{
  Find one of the nine specified macroblocks that yields minimum MAD;
  if offset = 1 then last = TRUE;
  offset = [offset/2];
  Form a search region with the new offset and new center found;
}
END

```

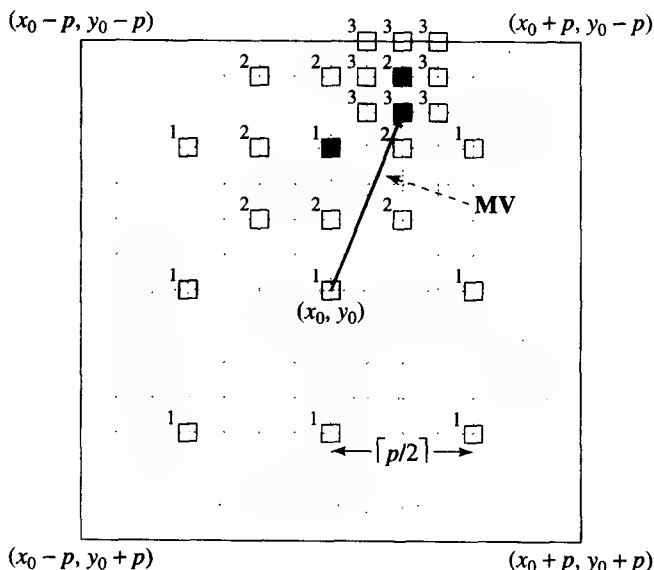


图 10-2 运动向量的 2D 对数搜索

在顺序搜索中,需要同参考帧进行  $(2p+1)^2$  次宏块间的比较,而在 2D 对数搜索中只需进行  $9 \cdot (\lceil \log_2 p \rceil + 1)$  次宏块间比较。实际上,应该是  $8 \cdot (\lceil \log_2 p \rceil + 1) + 1$  次,因为上一次迭代所生成的最小 MAD 在下次迭代中可以直接使用,无需再次计算。因此,计算复杂度降为  $O(\log p \cdot N^2)$ 。由于  $p$  通常情况和  $N$  是一个数量级的,所以与  $O(p^2 N^2)$  相比,已经得到非常明显的改善。

采用上一节例子中的数据,每秒的运算量将为:

$$\begin{aligned}
 \text{OPS\_per\_second} &= (8 \cdot (\lceil \log_2 p \rceil + 1) + 1) \cdot N^2 \cdot 3 \cdot \frac{720 \times 480}{N \cdot N} \cdot 30 \\
 &= (8 \cdot \lceil \log_2 15 \rceil + 9) \times 16^2 \times 3 \times \frac{720 \times 480}{16 \times 16} \times 30 \\
 &\approx 1.25 \times 10^9.
 \end{aligned}$$

### 10.3.3 分层搜索

运动向量的搜索采用分层的方法(多分辨率)也有诸多好处,在该方法中,初始的运动向量估计是从显著降低分辨率后的图像中获得的。图 10-3 描述了一个 3 层的搜索算法,原始图像为第 0 层,第 1 层和第 2 层的图像是通过将上一层图像的分辨率减半而获得的。初始的搜索从第 2 层开始。由于宏块变小了,  $p$  值也随之正比例的减小,这一层的计算量也大大的缩减(减小为原来的  $1/16$ )。

由于图像的分辨率低以及缺少图像细节内容,所以初始的运动向量估计值是比较粗糙的。

但是这个值会一层层地进行修正,直到第0层。假设第 $k$ 层运动向量的估计值为 $(u^k, v^k)$ ,那么第 $k-1$ 层将以 $(2 \cdot u^k, 2 \cdot v^k)$ 为中心,在 $3 \times 3$ 的区域内进行搜索,从而修正运动向量的估计值。换句话说,第 $k-1$ 层对运动向量的修正必须使修正后的运动向量值 $(u^{k-1}, v^{k-1})$ 满足

$$(2u^k - 1 \leq u^{k-1} \leq 2u^k + 1, 2v^k - 1 \leq v^{k-1} \leq 2v^k + 1)$$

并得到该宏块的最小MAD。

设 $(x_0^k, y_0^k)$ 代表目标帧中第 $k$ 层某宏块的中心。目标帧的中心为 $(x_0^0, y_0^0)$ 的宏块的分层运动向量搜索算法的过程如下:

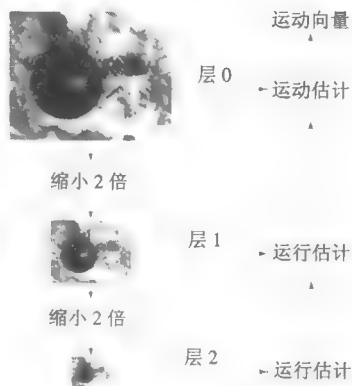


图 10-3 一个3层的运动向量搜索

### 程序 10-3 次运动向量: 分层搜索

```

BEGIN
// Get macroblock center position at the lowest resolution level k, e.g., level 2.
 $x_0^k = x_0^0 / 2^k$ ;  $y_0^k = y_0^0 / 2^k$ ;
Use Sequential (or 2D Logarithmic) search method to get initial estimated  $MV(u^k, v^k)$ 
at level k;
WHILE last  $\neq$  TRUE
|
|   Find one of the nine macroblocks that yields minimum MAD
|   at level  $k - 1$  centered at
|    $(2(x_0^k + u^k) - 1 \leq x \leq 2(x_0^k + u^k) + 1, 2(y_0^k + v^k) - 1 \leq y \leq 2(y_0^k + v^k) + 1)$ ;
|   if  $k = 1$  then last = TRUE;
|    $k = k - 1$ ;
|   Assign  $(x_0^k, y_0^k)$  and  $(u^k, v^k)$  with the new center location and motion vector;
|
END

```

我们将使用前面相同的例子来计算在三层的分层搜索中每秒所需要的计算量。为了简便起见,初始化时生成多分辨率的目标帧和参考帧的计算将不考虑在内,并且我们假设每一层的搜索中采用顺序搜索的办法。

同样,每秒处理的宏块数仍为 $\frac{720 \times 480}{N \cdot N} \times 30$ 。但处理每一个宏块所需要的计算量减少为:

$$\left[ \left( 2 \left\lceil \frac{p}{4} \right\rceil + 1 \right) \left( \frac{N}{4} \right)^2 + 9 \left( \frac{N}{2} \right)^2 + 9N^2 \right] \times 3$$

因此

$$\begin{aligned}
 \text{OPS\_per\_second} &= \left[ \left( 2 \left\lceil \frac{p}{4} \right\rceil + 1 \right) \left( \frac{N}{4} \right)^2 + 9 \left( \frac{N}{2} \right)^2 + 9N^2 \right] \\
 &\quad \times 3 \times \frac{720 \times 480}{N \cdot N} \times 30 \\
 &= \left[ \left( \frac{9}{4} \right)^2 + \frac{9}{4} + 9 \right] \times 16^2 \times 3 \times \frac{720 \times 480}{16 \times 16} \times 30 \\
 &\approx 0.51 \times 10^9
 \end{aligned}$$

表 10-1 总结了这三种运动向量搜索方法在视频图像的分辨率为  $720 \times 480$ ，帧率为 30， $p$  值分别为 15 和 7 时的性能优劣。

表 10-1 运动向量搜索方法计算代价的对比

搜索方法	OPS_per_second for $720 \times 480$ at 30 fps	
	$p=15$	$p=7$
顺序搜索	$29.89 \times 10^9$	$7.00 \times 10^9$
2D 对数搜索	$1.25 \times 10^9$	$0.78 \times 10^9$
3 层次搜索	$0.51 \times 10^9$	$0.40 \times 10^9$

## 10.4 H.261

H.261 是一种早期的数字电视压缩标准。因为它的基于运动补偿的压缩思想在后来所有的压缩标准中仍然采用，所以我们将首先讨论 H.261。

国际电报电话咨询委员会 (CCITT) 在 1988 年提出了 H.261 标准，该标准于 1990 年被国际电信联盟电信标准组织 ITU-T (CCITT 的前身) 所采纳[2]。

这个标准是为了在 ISDN 上进行可视电话、视频会议和提供其他视听服务而制定的。最初，希望它能支持多个 (1~5 个) 384kps 的信道。但是，视频编码器只提供  $p \times 64$  kbps 的码率 ( $p$  的取值范围为 1~30)。因此，该标准也称作  $p \times 64$  标准，读作“ $p$  星 64”。该标准要求视频编码器的延迟必须低于 150ms，以便于视频能够用于实时的双向视频会议。

H.261 属于下列 ITU 为可视电话系统推荐的一系列标准：

- H.221 支持 64kbps~1920kbps 的视听信道的帧格式。
- H.230 视听系统中的帧控制信号。
- H.242 视听通信协议。
- H.261 速率为  $p \times 64$  kbps 的用于视听服务的视频编码/解码器。
- H.320 传输率为  $p \times 64$  kbps 的窄带视听终端标准。

表 10-2 列出了 H.261 支持的视频格式。H.261 中的色度二次采样采用了 4:2:0 的格式。考虑到当时网络通信的能力比较差，指定 H.261 必须支持 CCIR 的 QCIF，而对 CIF 的支持是可选的。

表 10-2 H.261 支持的视频格式

视频格式	亮度图像分辨率	色度图像分辨率	码率 (Mbps) (若 30fps 或未压缩)	H.261 支持
QCIF	$176 \times 144$	$88 \times 72$	9.1	必须
CIF	$352 \times 288$	$176 \times 144$	36.5	可选

图 10-4 描述了一个典型的 H.261 帧序列。在这里定义了两种类型的图像帧：I 帧 (intra-frames) 和 P 帧 (inter-frames)。

I 帧被视为独立的图像。基本上，在每一个 I 帧内应用和 JPEG 相似的变换编码方法，因此被称作“intra”。

P 帧不是独立的。它们采用的是前向预测的编码方法，在该方法中当前宏块是通过先前的 I 帧或者 P 帧中相似的宏块预测出来的，并对宏块间的差 (difference) 进行编码。因此，P 帧的编

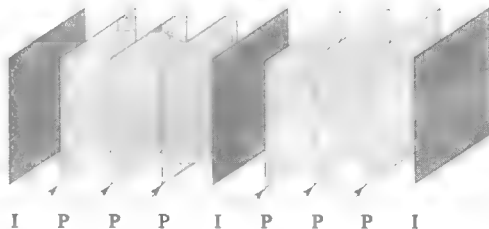


图 10-4 H.261 的帧序列

码中包含时间冗余消除，而 I 帧的编码只能对空间冗余进行消除。要记住，从一个先前的 P 帧来进行预测是允许的（不只是从先前的 I 帧进行预测）。

296 两个 I 帧之间的间隔是可变的，这是由编码器所决定的。通常，一个普通的数字视频每秒有一对 I 帧。H.261 标准中运动向量的测量是以整个像素为单位的，限制范围为 $\pm 15$  像素即  $p$  值为 15）。

10.4.1 I 帧编码

宏块是原图 Y 帧中  $16 \times 16$  的像素块。因为采用了 4:2:0 的色度二次采样，所以在 Cb 帧和 Cr 帧中，对应为  $8 \times 8$  大小的区域。因此，一个宏块由 4 个 Y 块，1 个 Cb，1 个 Cr 和  $8 \times 8$  的块组成。

对每一个  $8 \times 8$  的子块，都要进行离散余弦变换。同 JPEG 算法（在第 9 章中详细讨论过）一样，离散余弦变换后的 DCT 系数也需要进行量化。最后，通过 Z 字扫描并进行熵编码（如图 10-5 所示）。



图 10-5 I-帧编码

10.4.2 P 帧预测编码

图 10-6 是 H.261 中基于运动补偿的 P 帧编码方案。对目标帧中的每一个宏块来说，我们通过前面讨论的 3 种方法中的任意一种进行运动向量的分配。接着，再用差值宏块测量预测误差（prediction error）。同样，宏块由 4 个 Y 子块、一个 Cb 子块和一个 Cr 子块组成。这些  $8 \times 8$  的子块都需要经过离散余弦变换、量化、Z 字扫描和熵编码四个步骤。而且，运动向量也需要编码。

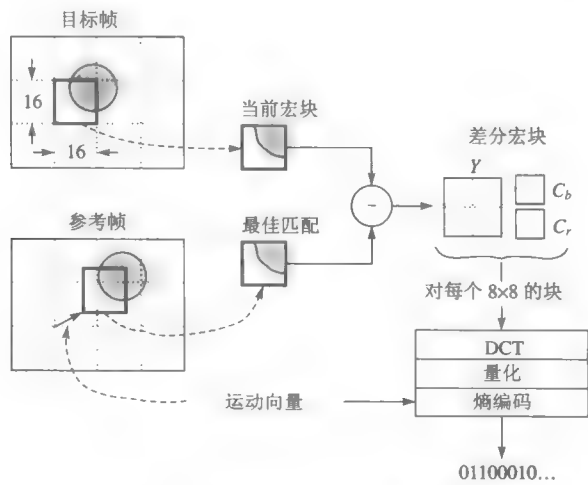


图 10-6 H.261 中基于运动补偿的 P 帧编码

有时, 预测误差超过了一个我们可以接受的水平, 因此找不到一个更好的匹配。在这种情况下, 就需要将宏块本身进行编码(当作一个 I 帧), 该宏块也称为未进行运动补偿的宏块(non-motion-compensated macroblock)。

P 帧编码是将宏块间的差值进行编码, 而不是对目标宏块本身进行编码。因为宏块间差值的熵通常比目标宏块的熵小很多, 所以通过这样的方法可以得到一个比较大的压缩率。

实际上, 运动向量也不是直接编码的, 而是将前一个宏块的运动向量和当前宏块运动向量的差值 MVD 送到编码器进行编码:

$$\text{MVD} = \text{MV}_{\text{Preceding}} - \text{MV}_{\text{Current}} \quad (10.3)$$

### 10.4.3 H.261 的量化

H.261 标准中的量化没有像 JPEG 和 MPEG 那样采用  $8 \times 8$  量化矩阵, 而是对一个宏块中所有的 DCT 系数均采用一个常数, 称为步长 (step-size)。

根据需要 (例如视频的码率控制), 步长可以取 2~62 中的任何一个偶数值, 共 31 种取值。但是有一个例外, 帧内编码中 DC 系数总是采用 8 作为步长。如果使用 DCT 和 QDCT 表示量化前后的 DCT 系数, 那么帧内编码中的 DC 系数是:

$$\text{QDCT} = \text{round}\left(\frac{\text{DCT}}{\text{step\_size}}\right) = \text{round}\left(\frac{\text{DCT}}{8}\right) \quad (10.4)$$

其他的系数是:

$$\text{QDCT} = \left\lfloor \frac{\text{DCT}}{\text{step\_size}} \right\rfloor = \left\lfloor \frac{\text{DCT}}{2 \times \text{scale}} \right\rfloor \quad (10.5)$$

其中 scale 是 [1, 31] 上的一个整数。

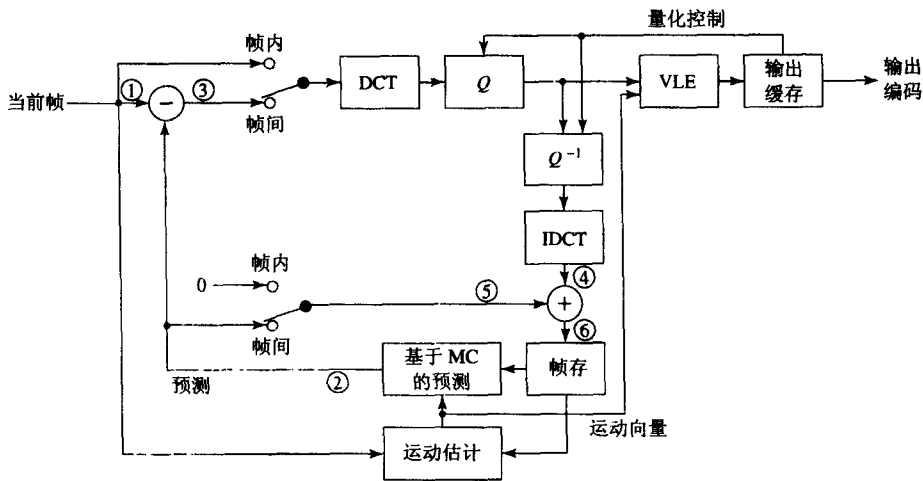
在 8.4.1 节中讨论过的中宽量化器中, 通常采用四舍五入的方式进行量化 (使用 round 运算符)。式 10.4 使用的就是这种量化器。但式 10.5 使用的是 floor 操作符, 因此在量化空间中留下一个中心死区 (如图 9-8 所示), 其中很多值都被映射为 0。

### 10.4.4 H.261 编码器和解码器

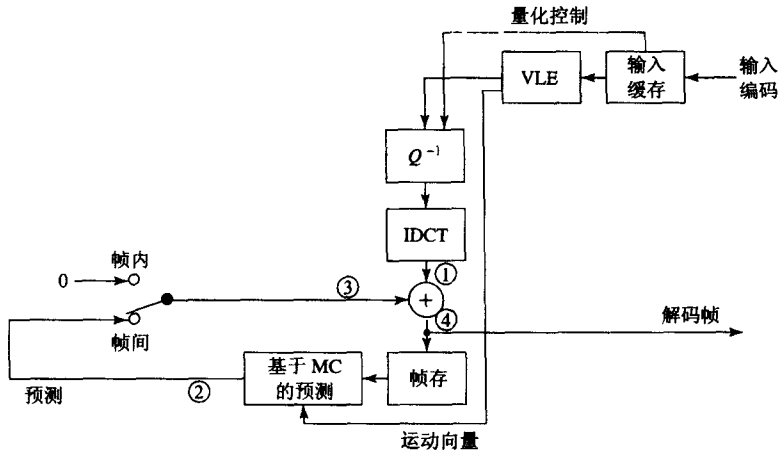
图 10-7 分别说明了 H.261 编码器和解码器工作的全过程。在这里,  $Q$  和  $Q^{-1}$  分别代表量化和其逆过程。帧内编码和帧间编码模式可以通过一个多路转换器进行切换。要避免编码误差的传播,

- 通常视频中每秒将一个 I 帧发送两次。
- 如前面所讨论的 (见 6.3.5 节中的 DPCM), 解码帧 (不是原始帧) 在运动估计中作为参考帧。

为了分析编码器和解码器的工作过程, 在这里我们引入了一个  $I$ 、 $P_1$  和  $P_2$  三帧先进行编码再解码的场景。在表 10-3 和表 10-4 中列出了经过观察点的数据变化情况, 观察点在图 10-7 中用画圈的数字表示。我们把  $I$ 、 $P_1$  和  $P_2$  当作初始数据,  $\tilde{I}$ 、 $\tilde{P}_1$  和  $\tilde{P}_2$  是解码后的数据 (与原数据相比是有损耗的),  $P'_1$  和  $P'_2$  是帧间编码中的预测图像。



a) 编码器



b) 解码器

图 10-7 H.261

表 10-3 H.261 编码器上观察点处的数据流

当前帧	观察点					
	1	2	3	4	5	6
I	I		$\tilde{I}$	0	$\tilde{I}$	
P <sub>1</sub>	P <sub>1</sub>	P <sub>1</sub>	D <sub>1</sub>	$\tilde{D}_1$	P <sub>1</sub>	$\tilde{P}_1$
P <sub>2</sub>	P <sub>2</sub>	P <sub>2</sub>	D <sub>2</sub>	$\tilde{D}_2$	P <sub>2</sub>	$\tilde{P}_2$

表 10-4 H.261 解码器上观察点处的数据流

当前帧	观察点			
	1	2	3	4
I	$\tilde{I}$		0	$\tilde{I}$
P <sub>1</sub>	$\tilde{D}_1$	P <sub>1</sub>	P <sub>1</sub>	$\tilde{P}_1$
P <sub>2</sub>	$\tilde{D}_2$	P <sub>2</sub>	P <sub>2</sub>	$\tilde{P}_2$

在编码器中，当前帧为 I 帧时，观察点 1 从 I 帧中接收宏块，在表 10-3 中用 I 表示。每一个 I 经过离散余弦变换、量化、熵编码，将结果放入输出缓冲区中，准备发送。

同时，I 量化后的 DCT 系数被送到  $Q^{-1}$  和 IDCT 模块进行量化逆变换和逆离散余弦变换，把观察点 4 得到的数据记为  $\tilde{I}$ 。把观察点 5 的 0 输入和观察点 4 的数据相加，观察点 6 仍然得到数据  $\tilde{I}$ ，并保存在帧内存中，用于下一帧 P<sub>1</sub> 的运动估计和基于运动补偿的预测。

量化控制就是反馈控制，即当输出缓冲区快要占满时，量化步长就要增加，以减少编码数据



的大小。这个过程叫编码率控制过程 (encoding rate control process)。

当接下来的当前帧  $P_1$  到达观察点 1 时, 立即调用运动估计过程, 在帧  $\tilde{I}$  中为  $P_1$  中的每一个宏块寻找最匹配的宏块, 求得运动向量。这个运动向量的估计值同时被送到基于运动补偿的预测器 (MC-based prediction) 和可变长度编码器 (VLC)。基于运动补偿的预测器给出  $P_1$  中最匹配的宏块, 在观察点 2 用  $P'_1$  表示。

在观察点 3, 得到预测误差, 其值为  $D_1 = P_1 - P'_1$ 。接着,  $D_1$  经过离散余弦变换、量化和熵编码, 将结果送入输出缓冲区中。和前面一样,  $D_1$  中离散余弦变换的参数被送至  $Q^{-1}$  和 IDCT 模块进行量化逆变换和逆离散余弦变换, 在观察点 4 得到  $\tilde{D}_1$ 。

在观察点 6,  $\tilde{D}_1$  和  $P'_1$  相加, 得到  $\tilde{P}_1$ , 并存放在帧内存中, 用于下一帧  $P_2$  的运动估计和基于运动补偿的预测。 $P_2$  的编码过程和  $P_1$  的编码过程非常相似, 只是当前帧变为  $P_2$ , 而  $P_1$  变为参考帧。

在解码器中, 输入的数据首先会经过熵解码、量化逆变换和逆离散余弦变换三个过程。对于 I 帧, 解码后的数据首先出现在观察点 1, 接着是在观察点 4, 用  $\tilde{I}$  表示。 $\tilde{I}$  被当作第一个输出帧, 并且同时送到帧内存中进行保存。

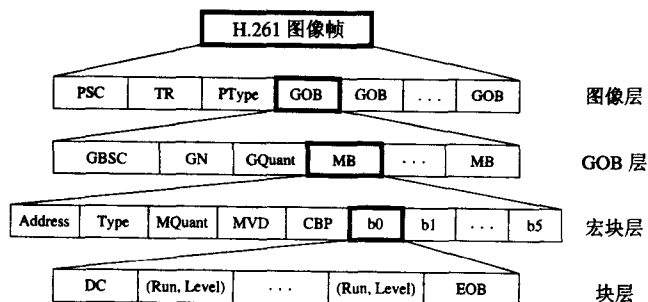
随后,  $P_1$  作为帧间输入码进行解码, 在观察点 1 得到预测误差  $\tilde{D}_1$ 。由于当前宏块的运动向量也进行了熵解码且被送往基于运动补偿的预测器, 所以相应的预测宏块  $P'_1$  可以在  $\tilde{I}$  中找到, 并在观察点 2、3 得到结果。在观察点 4,  $P'_1$  和  $\tilde{D}_1$  相加得到  $\tilde{P}_1$ 。 $\tilde{P}_1$  作为解码后的帧输出, 同时存放在帧内存中。 $P_2$  的解码过程和  $P_1$  的解码过程相似。

#### 10.4.5 H.261 视频位流语法概述

下面简要介绍 H.261 视频位流的语法 (见图 10-8)。这是一个分层的结构, 共有四层: 图像层 (Picture Layer)、块组层 (Group of Block Layer, GOB)、宏块层 (Macro Block Layer, MB) 和块层 (Block Layer)。

##### 1. 图像层

图像起始码 (Picture Start Code, PSC) 标记了图像之间的界限。时间参考 (Temporal Reference, TR) 提供该图像的时间戳。由于有时进行时间二次采样而导致有些图像不能被发送, 所以 TR 在保持视频和音频的同步中是非常重要的。PType (图像类型) 指定该图像的格式, 如 CIF 或 QCIF。



PSC	图片开始码	TR	时间参照
PType	图片类型	GOB	块组
GBSC	GOB 开始码	GN	组号
GQuant	GOB 量化器	MB	宏块
MQuant	MB 量化器	MVD	运动向量数据
CBP	编码块模式	EOB	块结束

图 10-8 H.261 视频码流中的语法

2. 块组层

在 H.261 标准中, 图像被分割为 11×3 大小的宏块 (即在亮度图像中为 176×48 像素大小) 的区域, 每一个区域称为一个块组 (GOB)。图 10-9 描述了 CIF 和 QCIF 亮度图像中块组的排列。例如, 在 CIF 图中有 2×6 个块组, 相应图像的分辨率为 352×288。

每一个块组都有自己的起始码 (GBSC) 和组号 (GN)。起始码是唯一的, 不必对整个位流中的变长编码进行解码就可以识别出起始码。在网络发生错误导致一个比特的数据错误或者一些数据丢失的情况下, 采用 H.261 标准的视频可以在下一个可识别的块组到来时进行数据恢复和重新同步, 从而避免错误的传播。

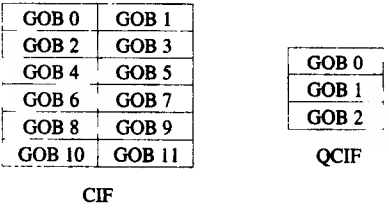


图 10-9 H.261 亮度图的块组设置

GQuant 表示在组块中将使用的量化器, 除非被后来的宏块量化器 (Macroblock Quantizer, MQuant) 所取代。GQuant 和 MQuant 在式 10.5 中由 scale 表示。

3. 宏块层

每一个宏块 (MB) 都有自己的地址, 指明它在组块中的位置。每个宏块还包括所采用的量化器 (MQuant) 以及六个 8×8 的图像子块 (4Y, 1Cb, 1Cr)。类型 (Type) 指明该块是帧内块还是帧间块, 有没有运动补偿等。运动向量的数据 (MVD) 是通过求先前宏块和当前宏块之间的差值而得到的。此外, 由于在运动估计中, 某些宏块得到了很精确的估计, 而有些估计效果却很差, 所以用一位掩码 (Coded Block Pattern, CBP) 表示该信息。只有估计值精确的块才传送其 DCT 系数。

4. 块层

对于每一个 8×8 的图像子块来说, 位流都是从 DC 值开始, 接着是成对的 AC 的 0 游长 (Run) 和紧接的非 0 值 (Level), 最后是结束符 (EOB)。“Run” 取值范围为 [0, 63]。“Level” 反映出量化值, 范围为 [-127, 127], 且 Level≠0。

10.5 H.263

H.263 是一个经过改进的视频编码标准[3], 主要用于公共电话交换网络 (PSTN) 上的视频会议或其他可视化服务传输。它旨在以尽可能低的 (64kbps 以下) 码率进行通信。H.263 在 1995 年被 ITU-T 所采纳。和 H.261 相似, 它在帧间编码中采用预测编码来减少时间冗余信息, 对剩下的信号采用变换编码来减少空间冗余信息 (如帧内编码及帧间预测的差分宏块)。

除了 CIF 和 QCIF 之外, H.263 还支持 sub-QCIF、16CIF 和 16CIF 格式的视频。表 10-5 总结了 H.263 支持的视频格式。如果不进行压缩并且设定 30fps 的帧率, 高分辨率的视频 (如 16CIF) 所占用的带宽将非常大 (>500Mbps)。对于压缩的视频来说, 标准规定了每幅图像的最大码率 (BPPmaxKb), 单位是 1024b (1K)。实际上, 压缩后的 H.263 视频可以达到较低的码率。

表 10-5 H.263 支持的视频格式

视频格式	亮度图像分辨率	色度图像分辨率	码率 (Mbps) (如果是 30fps 或未压缩)	码率 (kbps) BPPmaxKb (压缩)
Sub_QCIF	128×96	64×48	4.4	64
QCIF	176×144	88×72	9.1	64
CIF	352×288	176×144	36.5	256
4CIF	704×576	352×288	146.0	512
16CIF	1408×1152	704×576	583.9	1024

和 H.261 相同, H.263 标准同样支持组块的概念。不同的是, H.263 中组块 (GOB) 没有固定的大小, 它们总是起止于图像的左右边界。如图 10-10 所示, 每一个 QCIF 亮度图像包括 9 个组块, 每一个组块大小为  $11 \times 1\text{MB}$  ( $176 \times 16$  像素); 而每一个 4CIF 亮度图像包括 18 个组块, 每一个组块大小为  $44 \times 2\text{MB}$  ( $704 \times 32$  像素)。

303

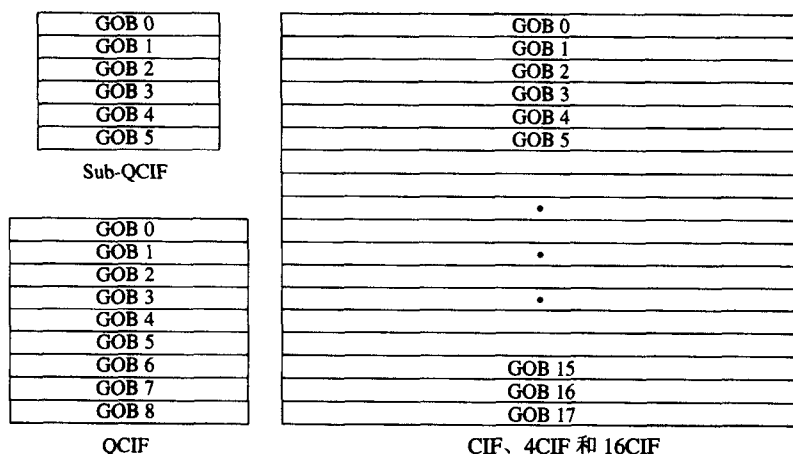
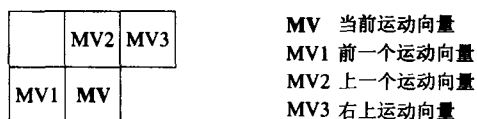


图 10-10 H.263 亮度图中的块组的设置

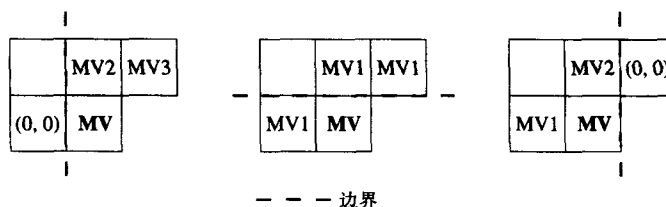
### 10.5.1 H.263 的运动补偿

H.263 中的运动补偿过程和 H.261 中的运动补偿过程相似。但是, 运动向量 (MV) 不只是从当前宏块产生的。MV 的水平分量和垂直分量分别由当前宏块左方、上方和右上方宏块的运动分量 (MV1, MV2, MV3) 的水平分量和垂直分量的平均值预测得出 (见图 10-11a)。也就是说, 宏块的运动向量 MV ( $u, v$ ) 为,

$$\begin{aligned} u_p &= \text{median}(u_1, u_2, u_3), \\ v_p &= \text{median}(v_1, v_2, v_3). \end{aligned} \quad (10.6)$$



a) 当前宏块的预测运动向量是 (MV1, MV2, MV3) 的中值



b) 当前宏块是图像或者块组的边缘处时, 指定运动向量的特殊方法

图 10-11 H.263 的运动向量预测

在 H.263 中,并不是对  $MV(u, v)$  进行编码,而是对误差向量  $(\delta u, \delta v)$  进行编码,其中  $\delta u = u - u_p$ ,  $\delta v = v - v_p$ 。如图 10-11b 所示,如果当前宏块在图像或组块的边界时,使用  $(0, 0)$  或者  $MV1$  作为边界外宏块的运动向量。

为了改善运动补偿的效果,也就是说减少预测的误差, H.263 支持半像素精度 (half-pixel precision) 的预测,而 H.261 只支持完整像素精度的预测。 $MV(u, v)$  中水平向量  $u$  和垂直向量  $v$  的默认取值范围为  $[-16, 15.5]$ 。

半像素位置的像素值是通过双线性插值 (bilinear interpolation) 方法得到的,如图 10-12 所示。

图中 A、B、C、D 和 a、b、c、d 分别代表全像素位置和半像素位置,“/”表示整除。

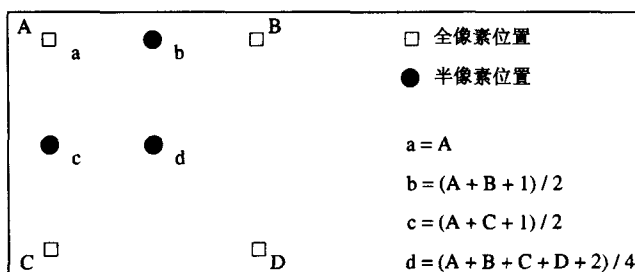


图 10-12 H.263 通过双线性插值进行半像素精度的预测

### 10.5.2 H.263 的可选编码模式

除了核心编码算法外, H.263 在附件中指定了许多可选的编码方法。四个常用的算法如下:

#### • 无限制的运动向量模式

参考的像素不再限制在图像的边界内。当运动向量指向图像边界时,将使用边界上几何位置最靠近参考像素的像素点的值。当图像内容在边界上移动时这种方式是非常有利的,比如物体的移动或者摄像头的移动。该模式也允许运动向量取值范围的扩展。运动向量取值的最大范围为  $[-31.5, 31.5]$ ,这使得视频中快速运动物体的编码变得十分有效。

#### • 基于语法的算术编码模式

与 H.261 相同, H.263 使用变长编码方法作为 DCT 系数的默认编码方法。变长编码意味着每一个符号必须编码成一个固定的完整的字节。采用算术编码,这种限制被取消,而且可以达到更高压缩率。实验表明,使用该方法,帧间编码可以节省 4% 的码率,帧内编码可以节省 10% 左右的码率。

和 H.261 相似, H.263 的语法由一个四层的结构构成,每一层结合定长或者变长方式进行编码。在基于语法的算术编码 (Syntax-based Arithmetic Coding, SAC) 模式中,所有可变长度的编码操作算术编码操作替换。根据每一层的语法,算术编码器需要将多个分量编码成不同的位流。因为每个位流都有不同的分布,所以 H.263 为每一个分布定义了一个模型,并且算术编码器在空闲的时候根据语法进行模型切换。

#### • 高级预测模式

该模式下,运动补偿中宏块的大小从 16 减小到 8。在亮度图中,每一个宏块产生四个运动向量(从每一个  $8 \times 8$  的块)。然后,  $8 \times 8$  亮度预测块中的每一个像素都是由三个预测值的加权和得到的,一个是基于当前亮度块的运动向量,另外两个是当前块四个邻居块中两个块的运动向量,即一个是当前亮度块的左邻块或右邻块,另一个是上邻块或下邻块。尽管传送四个运动向量会带来额外的开销,但是这种方式预测的准确度较高,因此在数据压缩方面有很好的效果。

#### • PB 帧模式

和在 MPEG (在下一章中详细讨论) 中一样,我们引入了 B 帧,它是同时由先前帧和后一帧双向预测而得到的。它可以改善预测的质量,因此可以在不牺牲画质的条件下提高压缩率。在 H.263 中,一个 PB 帧是由两个编码为一个单元的图片组成:一个是 P 帧,从先前解码的 I 帧或者 P 帧(或者 PB 帧的 P 帧部分)预测得到;一个是 B 帧,同时从先前解码的 I 帧或者 P 帧以及当

前正在编码的 P 帧预测得到 (参见图 10-13)。

PB 帧模式在 PTYPE 中使用。因为 P 帧和 B 帧在 PB 帧中紧密耦合, 所以 B 帧中的双向运动向量不需要独立生成。相反, 它们可以首先得到一个暂时的值, 再通过 P 帧[4]的向前运动向量进行修正, 这样可以减少 B 帧上的码率的开销。PB 帧模式在视频图像变化不太大时可以得到满意的效果。在视频图像变化比较大的情况下, PB 帧模式不如 B 帧模式的压缩效果好。在 H.263 的版本 2 中提出了一个增强型的模式。

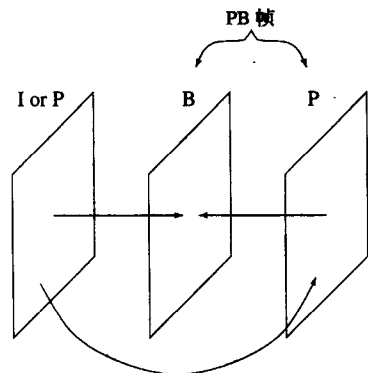


图 10-13 H.263 中一个 PB 帧

### 10.5.3 H.263+和 H.263++

H.263 的第二个版本 (称作 H.263+) 在 1998 年被 ITU-T Study Group 16 采纳。它完全兼容 H.263 版本 1 中的所有设计。

提出 H.263+ 的目的是为了拓展潜在的应用, 并在定制的源的格式、不同像素尺寸比和时钟频率方面增加灵活性。H.263+ 包括多种改善编码效率和差错恢复的建议[5]。同时, 除了 H.263 中的四种可选编码模式之外, 它还提供了 12 种全新的可选的模式。

由于 H.263+ 是在 MPEG-1 和 MPEG-2 之后开发的, 所以其中吸收了很多 MPEG 标准的内容。下面我们简单地介绍其中的一些内容, 细节的讨论在下一章进行。

- 在 H.263+ 中, 重新定义了无限制的运动向量模式。它采用可逆变长编码方法 (Reversible Variable Length Coding, RVLC) 对运动向量的差进行编码。RVLC 编码器能进行正向和逆向解码, 这样就将传输误差的影响减至到最小。运动向量的取值范围扩大到  $[-256, 256]$ 。

RVLC 构造的细节可以参考[6, 7]。

- 用一个宏块片 (slice) 结构替代组块, 以达到更大的灵活性。一个宏块片可以包括可变数目个宏块。传输顺序可以是顺序的也可以是任意的, 并且宏块片的形状也不一定是矩形的。
- H.263+ 实现了时间、空间、信噪比上的可伸缩性。可伸缩性指对多种约束, 如显示分辨率、带宽、硬件功能的处理能力。用于时间可伸缩性的增强层通过在两个 P 帧间插入 B 帧来提高预测质量。

信噪比可伸缩性是通过使用步长越来越小的量化器来把附加的增强层编码到位流中来实现的。因此, 解码器可以根据计算和网络限制来决定需要解码的层数。空间可伸缩性的概念和信噪比可伸缩性的概念相似。在这种情况下, 增强层提供增加的空间分辨率。

- H.263+ 支持改善的 PB 帧模式。和版本 1 不同, B 帧中的两个运动向量不必由前一个 P 帧的运动向量得到。相反, 它们可以像 MPEG-1 和 MPEG-2 中那样独立生成。
- 解决过滤器在循环编码过程中可以减少阻塞带来的影响。这个过滤器用在四个亮度块和两个色度块的边缘处。系数的权重依赖于块量化器的步长。利用这种技术, 不仅能取得很好的预测效果, 而且减少了人为的块生成。

在版本 2 之后, H.263 仍在继续发展, 新的版本称作 H.263++[8]。H.263++ 包括 H.263 中的基线编码方法, 以及增强型的参考图片选择 (Enhanced Reference Picture Selection, ERPS)、数据划分块 (Data Partition Slice, DPS) 和增强信息的附加方面的建议。

ERPS 模式通过管理一个用于存放图像帧的多帧的缓冲区来运行, 可以提高编码效率和错误恢复能力。DPS 模式通过把头数据和运动向量从码率中的 DCT 系数中提取出来, 并且通过可逆的编码方式来保护运动向量, 来提供更强的错误恢复能力。附加的增强信息可以兼容 H.263。

## 10.6 进一步探索

Tekalp[9]和 Poynton[10]奠定了数字视频处理的基础。它们概述了解决视频中这类问题的数学基础。

Bhaskaran 和 Konstantinides[11]、Ghanbari[12]、Wang[13]等的书中对视频压缩算法进行了很好阐述, 并且提出了该问题中一些有趣的内容。

本书网站上第 10 章的 Further Exploration 中有一些关于 H.261、H.263 信息的链接, 包括:

- 教程和白皮书。
- 软件实现。
- H.263/H.263+图书馆。
- 基于 Java 的 H.263 解码器。

308

## 10.7 练习

1. 我收藏了很多 JPEG 图片 (在不同地方拍到的), 我决定把它们放到一个大的 H.261 压缩文件中, 以便整理和访问。我的想法是, 只使用一个浏览器, 把我收藏的所有图片整合在一起, 创建一个视频。讨论这个想法的可行性, 要考虑这些图片可以达到的压缩率。
2. 在基于块的视频编码中, 压缩或者解压缩哪个更有效? 简要说明原因。
3. 一个 H.261 的视频有三个颜色通道: Y、Cr 和 Cb。需要为每一个通道计算运动向量并进行传输吗? 证明你的答案。如果不是, 哪一个通道被用于运动补偿?
4. 回答下面用于运动向量的 2D 对数搜索的问题 (见图 10-14)。

目标 (当前) 帧为 P 帧。宏块的大小为  $4 \times 4$ 。运动向量是  $MV(\Delta x, \Delta y)$ , 其中  $\Delta x \in [-p, p]$ ,  $\Delta y \in [-p, p]$ 。在这个问题中, 假设  $p \equiv 5$ 。

帧中黑色的宏块左上角的坐标是  $(x_t, y_t)$ 。它包含 9 个黑色的像素, 每个像素的亮度值为 10;

其余 7 个像素点是背景的一部分, 统一亮度值为 100。参考帧 (前一帧) 有 8 个黑色像素点。

(a) 求  $\Delta x$ 、 $\Delta y$  的最优值, 宏块的平均绝对误差 (MAE) 是多少?

(b) 一步步地说明如何进行 2D 对数搜索, 包括搜索的位置和通道以及  $\Delta x$ 、 $\Delta y$  和 MAE 的所有中间值。

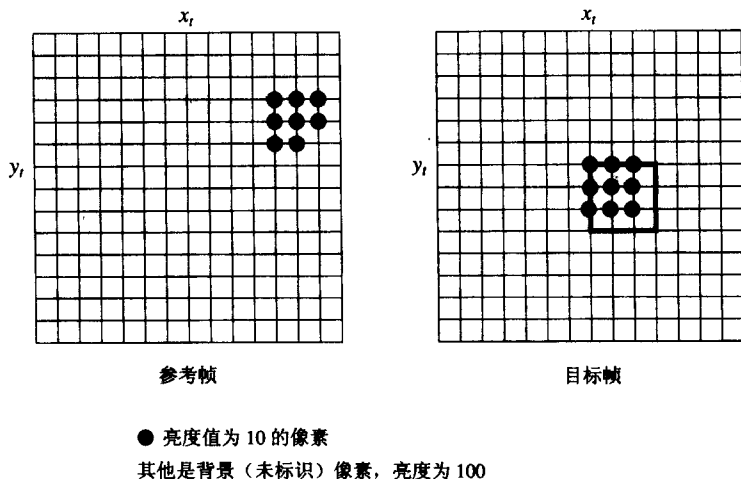


图 10-14 运动向量的 2D 对数搜索

5. 运动向量的对数搜索方法不是最优的, 因为它依赖于残余帧的连续性。  
(a) 解释为什么这个命题是正确的, 给出证明。

- (b) 给出一个该命题不成立的情况。
  - (c) 分层搜索算法也不是最优的吗?
6. 一个视频序列用 H.263 编码, 采用 PB 帧模式, 帧大小为 4CIF, 帧率为 30fps, 视频长度为 90 分钟。下面是已知的压缩参数: 平均每秒进行两次 I 帧编码; 达到质量要求的视频的 I 帧平均压缩率为 10:1, P 帧的压缩率平均是 I 帧的两倍; B 帧的压缩率是 P 帧的两倍。假设压缩参数包括所有必须的数据头, 计算压缩后视频的大小。
  7. 假设搜索窗口的大小为  $2p+1$ , 在 H.263 的高级预测模式下对 QCIF 视频进行运动评价的复杂性是多少? 使用:
    - (a) 顺序搜索
    - (b) 2D 对数搜索方法
    - (c) 分层搜索方法
  8. 讨论 H.263 中高级的预测模式是如何达到更好的压缩效果的。
  9. 在 H.263 的运动估计中, 将先前三个宏块 (参见图 10-11a) 运动向量的中值作为当前宏块的预测值。但是这样的中值不能很好给出最好的预测。描述在当前方法上可能进行的一些改善。
  10. H.263+中允许在 PB 帧中为 B 帧进行独立的前向 MV。和 H.263 的 PB 模式相比, 它们各有哪些优缺点。如果 B 帧有独立的运动向量, PB 联合编码的要点是什么?

## 10.8 参考文献

- [1] D. Marr, *Vision*, San Francisco: W. H. Freeman, 1982.
- [2] *Video Codec for Audiovisual Services at  $p \times 64$  kbit/s*, ITU-T Recommendation H.261, version 1 (Dec 1990), version 2 (Mar 1993).
- [3] *Video Coding for Low Bit Rate Communication*, ITU-T Recommendation H.263, version 1, (Nov 1995), version 2 (Feb 1998).
- [4] B.G. Haskell, A. Puri, and A. Netravali, *Digital Video: An Introduction to MPEG-2*, New York: Chapman & Hall, 1997.
- [5] G. Cote, B. Erol, and M. Gallant, "H.263+: Video Coding at Low Bit Rates," *IEEE Transactions on Circuits and Systems for Video Technology*, 8(7): 849-866, 1998.
- [6] Y. Takishima, M. Wada, and H. Murakami, "Reversible Variable Length Codes," *IEEE Transactions on Communications*, 43(2-4): 158-162, 1995.
- [7] C.W. Tsai and J.L. Wu "On Constructing the Huffman-Code-Based Reversible Variable-Length Codes," *IEEE Transactions on Communications*, 49(9): 1506-1509, 2001.
- [8] *Draft for H.263++ Annexes U, V, and W to Recommendation H.263*, International Telecommunication Union (ITU-T), 2000.
- [9] A.M. Tekalp, *Digital Video Processing*, Upper Saddle River, NJ: Prentice Hall PTR, 1995.
- [10] C.A. Poynton, *Digital Video and HDTV Algorithms and Interfaces*, San Francisco: Morgan Kaufmann, 2003.
- [11] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed., Boston: Kluwer Academic Publishers, 1997.
- [12] M. Ghanbari, *Video Coding: An Introduction to Standard Codecs*, London: Institute of Electrical Engineers, 1999.
- [13] Y. Wang, J. Ostermann, and Y.Q. Zhang, *Video Processing and Communications*, Upper Saddle River, NJ: Prentice Hall, 2002.

## 第 11 章 MPEG 视频编码 I: MPEG-1 和 MPEG-2

### 11.1 概述

运动图像专家组 (MPEG) 创立于 1988 年[1, 2], 主要负责为数字音频和视频的传输制定标准。其成员从 1988 年的 25 名专家发展到现在来自大约 200 个公司和组织的 350 人的一个团体[3, 4]。各公司都考虑到要在 MPEG 标准族中维护自己的利益, 因此, 他们仅仅定义了一个压缩位流, 也就是间接定义了解码器。而压缩算法和编码器是完全依赖于各生产商的。

本章我们将研究 MPEG-1 和 MPEG-2 中一些重要的设计问题。下一章将简要介绍后续标准 MPEG-4 和 MPEG-7 的一些基本原理以及不同点。

### 11.2 MPEG-1

MPEG-1 音频/视频数字压缩标准是由国际标准化组织/国际电子学委员会 (ISO/IEC) 下的 MPEG 组于 1991 年 11 月提出的, 用于至多对 1.5Mbit/s 的数字存储媒体运动图像及其伴音进行编码[5]。常见的数字存储媒体包括光盘 (CD) 和视频光盘 (VCD)。在 1.5Mbps 的数据传输率中, 1.2Mbps 用于已编码的视频, 256kbps 用于立体声。这样生成的图像质量相当于 VHS (家用视频系统) 的质量, 声音质量相当于 CD 的音频。

一般来说, MPEG-1 采用 CCIR601 数字电视格式, 这种格式也称为源输入格式 (Source Input Format, SIF)。MPEG-1 仅支持非隔行视频。通常, 对于 30fps 的帧速率, NTSC 制式视频的图片分辨率是 352×240, 对于 25fps 的帧速率, PAL 制式视频的图片分辨率是 352×288。采用 4:2:0 进行色度二次采样。

MPEG-1 标准 (也称为 ISO/IEC 11172 号建议[5]) 由五部分组成: 11172-1 系统、11172-2 视频、11172-3 音频、11172-4 MPEG 一致性和 11172-5 软件。简单地说, 在众多任务中, 系统负责将输出拆分成位流包, 进行多路传输以及将视频音频流同步。为了验证一个位流或解码器是否符合标准, 一致性部分详细描述了用于测试此问题的设计。软件部分包括 MPEG-1 标准解码器的完整软件实现以及一个编码器的软件实现实例。我们将对 MPEG-1 视频编码的主要特征进行分析, 对于 MPEG 音频编码的讨论我们留到第 14 章进行。

312

#### 11.2.1 MPEG-1 中的运动补偿

上一章讨论过, H.261 中的基于运动补偿的视频编码工作原理如下: 在运动估算中, 会为目标 P 帧的每个宏块分配一个从之前已编码的 I 帧或 P 帧的宏块中选出的与它最匹配的宏块, 这称为预测 (prediction)。当前宏块与匹配的宏块之间的差称为预测误差 (prediction error)。这个预测误差将被传送到 DCT 和接下来的编码步骤。

因为预测是从前面的帧得来的, 所以这种预测称为前向预测 (forward prediction)。由于在实际场景中会产生不可预测的移动和遮挡, 所以目标宏块与先前帧的宏块之间或许不能达到最佳匹配。图 11-1 描述了包含了半个球的当前帧的一个宏块与先前帧的宏块不能进行很好匹配的情况。因为球的一半被另一个物体遮挡住了。但是它从下一帧可以很容易的获得匹配。



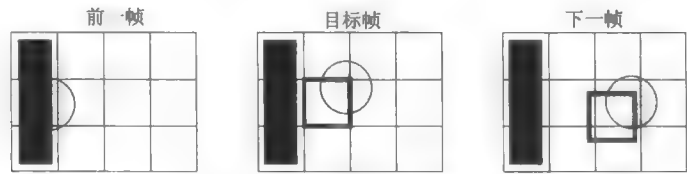


图 11-1 双向搜索的需求

MPEG 引入了第三类帧 (B 帧) 以及相应的双向运动补偿。图 11-2 描述了基于运动补偿的 B 帧编码原理。除了前向预测, 还用到了后向预测, 也就是此时用来进行匹配的宏块是从视频序列中未来的 I 帧或 P 帧中获得的。这样, B 帧的每一个宏块指定两个运动向量。一个由前向预测得来, 另一个由后向预测得来。

如果两个方向的匹配都成功, 那么两个运动向量都将被发送, 在与目标宏块进行比较产生预测误差之前, 将与两个相应的匹配宏块取平均 (图中用 “%” 表示)。如果只有一个参考帧的匹配是成功的, 那么只有一个运动向量及其相应的宏块会被使用, 或是前向预测的或是后向预测的。

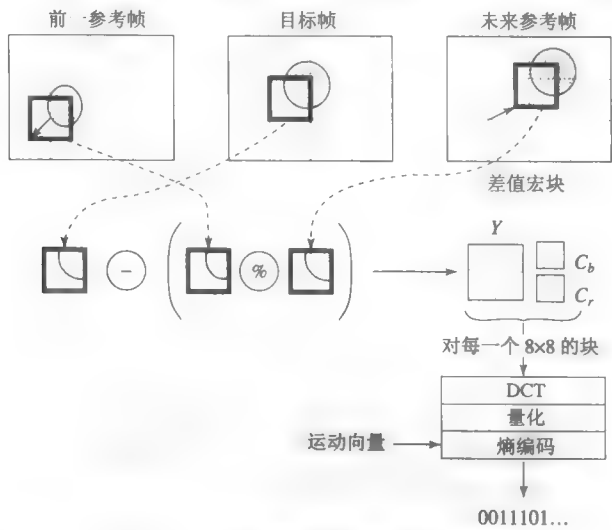


图 11-2 基于双向运动补偿的 B 帧编码

图 11-3 描述了视频帧一个可能的序列。实际帧的模式是在编码时决定的, 在视频头部有详细描述。MPEG 用  $M$  表示一个 P 帧和它之前的 I 帧或者 P 帧之间的间隔,  $N$  表示两个连续的 I 帧之间的间隔。在图 11-3 中,  $M=3$ ,  $N=9$ 。一个特例是当不使用 B 帧时,  $M=1$ 。

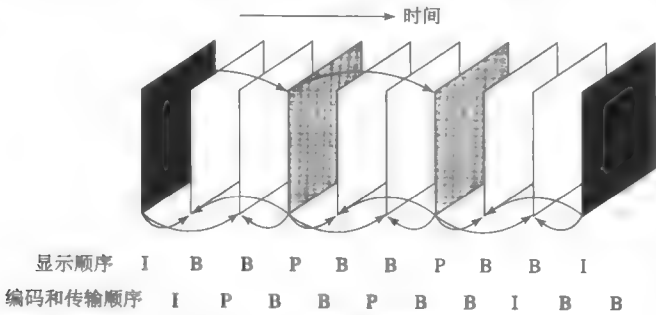


图 11-3 MPEG 帧序列

313  
314

由于 MPEG 编码器和解码器不能用于没有后续 P 帧或 I 帧的 B 帧宏块, 所以实际的译码和传输顺序 (图 11-3 下方所示) 与视频的显示顺序 (图 11-3 上方所示) 是不同的。因此, 在实时的网络传输中, 特别是在 MPEG 视频的传输中, 不可避免的延迟问题以及对缓冲区的需求都是极其重要的问题。

11.2.2 与 H.261 的其他主要区别

除了引入双向运动补偿 (B 帧) 外, MPEG-1 与 H.261 还有以下几个不同:

- **源格式** H.261 只支持 CIF (352×288) 和 QCIF (176×144) 两类源格式。MPEG-1 支持 SIF (在 NTSC 制式下为 352×240, 在 PAL 制式下为 352×288)。而且只要满足表 11-1 中的约束参数集 (Constrained Parameter Set, CPS), MPEG-1 还支持其他格式规范。

表 11-1 MPEG-1 的约束参数集

参 数	值
图片的水平尺寸	≤768
图片的垂直尺寸	≤576
每图片的宏块数	≤396
每秒的宏块数	≤9900
帧率	≤30fps
码率	≤1856 kbps

- **宏块片** 与 H.261 中的 GOB 不同, 一幅 MPEG-1 图片可以被分为一个或多个宏块片 (如图 11-4 所示), 这比 GOB 要灵活得多。只要可以填满整个图片, 它们可包含一幅图片中的可变数目的宏块并可以开始结束于任何位置。每个宏块片都独立编码。比如宏块片在量化器中可以有不同的缩放因子。这为码率控制提供了灵活性。

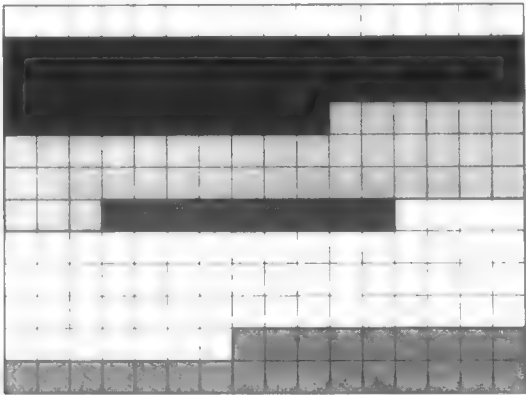


图 11-4 MPEG-1 图中的宏块片

315

而且, 宏块片的内容对于差错恢复是很重要的, 因为每个宏块片有唯一的宏块片起始码 (slice\_start\_code)。MPEG 中的宏块片类似于 H.261 (和 H.263) 中的 GOB: 它处于 MPEG 层次结构的最底层, 不需要将位流中整个变长编码组解码就可进行完全恢复。

- **量化** MPEG-1 的量化对于帧间编码和帧内编码采用不同的量化表 (见表 11-2 和表 11-3)。在一个宏块内, 用于帧内编码的量化器个数是不同的 (参见表 11-2)。这与 H.261 是不同的。对于 H.261, 一个宏块内用于 AC 系数的量化器个数是不变的。

表 11-2 帧内编码的默认量化表 ( $Q_1$ )

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

表 11-3 帧间编码的默认量化表 ( $Q_2$ )

16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

步长  $step\_size[i, j]$  的值由  $Q[i, j]$  与缩放因子的乘积决定, 其中  $Q_1$  或  $Q_2$  是上面的量化表之一, 缩放因子 (scale) 是 [1, 31] 内的一个整数。用 DCT 和 QDCT 表示量化前和量化后的 DCT 系数。帧内模式下的 DCT 系数为:

$$QDCT[i, j] = \text{round} \left( \frac{8 \times DCT[i, j]}{step\_size[i, j]} \right) = \text{round} \left( \frac{8 \times DCT[i, j]}{Q_1[i, j] \times scale} \right), \quad (11.1)$$

帧间模式下的 DCT 系数为:

$$QDCT[i, j] = \left\lfloor \frac{8 \times DCT[i, j]}{step\_size[i, j]} \right\rfloor = \left\lfloor \frac{8 \times DCT[i, j]}{Q_2[i, j] \times scale} \right\rfloor, \quad (11.2)$$

其中  $Q_1$  和  $Q_2$  分别指表 11-2 和表 11-3。

在式 (11.1) 中用到了 round 操作符, 因此没有留下任何无效区。然而, 在式 11.2 中由于采用了 floor 操作符, 在量化区域中会留下一个中心无效区。

- 为了增加基于运动补偿预测的精度, 减小预测误差, MPEG-1 允许运动向量具有半像素精度 (1/2 像素)。我们在 10.5.1 节中讨论的 H.263 的双线性插值 (bilinear interpolation) 技术可以用来产生在半像素位置所需的值。
- MPEG-1 支持 I 帧和 P 帧之间有较大的间距, 因此会有一个较大的运动向量搜索范围。与 H.261 运动向量的最大范围为  $\pm 15$  像素相比, MPEG-1 支持范围为  $[-512, 511.5]$  的半像素精度以及范围为  $[-1024, 1023]$  的精度为全像素的运动向量。然而, 由于图像分辨率实际情况的限制, 这个最大范围基本达不到。
- MPEG-1 位流允许随机访问。这已经在图片组 (Group of Pictures, GOP) 层实现了, 在图片组层中每一个图片组 (GOP) 都按时间编码。除此之外, 任一图片组的第一帧都是一个不依赖于其他帧的 I 帧。这样, GOP 层就可以使解码器在位流中寻找一个特定的位置并在那里开始解码。

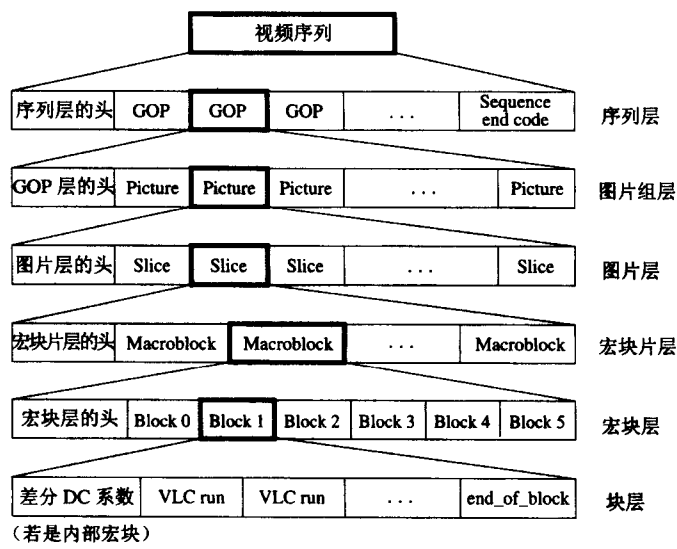
表 11-4 列出了所有类型的 MPEG-1 帧的典型大小 (单位为 KB)。可见, 压缩的 P 帧大小明显小于压缩的 I 帧, 因为帧间压缩采用了时间冗余。显然, B 帧比 P 帧更小, 一部分原因是采用了双向预测, 而且在考虑质量问题时 B 帧的优先级往往比较低。因此, 这样可以得到一个较高的压缩率。

表 11-4 MPEG-1 帧的典型压缩性能

类 型	大 小	压 缩 率
I	18KB	7 : 1
P	6KB	20 : 1
B	2.5KB	50 : 1
平均	4.8KB	27 : 1

11.2.3 MPEG-1 视频位流

图 11-5 描述了一个 MPEG-1 视频位流的六层结构。



1. 序列层

一个视频序列由一个或多个 GOP 构成。它通常以一个序列头开始。头中包含图片的信息，如图像水平大小 (horizontal\_size) 和垂直大小 (vertical\_size)、像素长宽比 (pixel\_aspect\_ratio)、帧率 (frame\_rate)、码率 (bit\_rate)、缓存大小 (buffer\_size) 以及量化矩阵 (quantization\_matrix) 等。各 GOP 间的可选序列头可以用来表明参数改变。

2. GOP 层

一个 GOP 包含一张或多张图片，其中的一张必为 I 图。GOP 头包含如时间码 (time\_code) 这样的信息，用来标识从序列开始到现在的时-分-秒-帧。

3. 图片层

之前讨论过，三种常见的 MPEG-1 图片的类型是 I 图 (帧内编码)、P 图 (预测编码) 和 B 图 (双向预测编码)。但还有一种不常见的类型——D 图 (DC 编码)，这种方法仅保留了 DC 系数。MPEG-1 中不允许将 D 图与其他类型的图混合，这样 D 图就行不通了。

4. 宏块片层

之前提到过，MPEG-1 引入了宏块片标识，用来进行码率控制以及在位丢失或受损后进行恢复和同步。宏块片可包含一张图片中的多个宏块。每个宏块片的长度和位置在头部指定。

5. 宏块层

每个宏块由四个 Y 块，一个  $C_b$  块，一个  $C_r$  块组成。所有的块都是 8x8 像素的。

6. 块层

如果块是帧内编码的，将先发送差分 DC 系数 (如 JPEG 中 DC 的 DPCM)，接着传送变长编码 (VLC)，作为 AC 系数。否则，DC 和 AC 系数全部都采用变长编码。

Mitchell et al[6]对不同的 MPEG-1 层头部的详细信息进行了描述。

317  
>  
318

### 11.3 MPEG-2

MPEG-2 标准的发展始于 1990 年。与 MPEG-1 这个为在一台计算机的 CD 上以较低的码率 (1.5Mbps) 存储和播放视频而制定的标准不同, MPEG-2[7]用于以高于 4Mbps 的码率存储和播放更高质量的视频。它最初是作为数字广播电视的标准而开发的。

在 20 世纪 80 年代后期,人们构想了一种高级电视 (Advanced TV, ATV),从而使高清晰度电视 (HDTV) 通过因特网广播开来。在 MPEG-2 的发展过程中,数字 ATV 在早期其他各种类似的解决 HDTV 的方法中占有很大优势。MPEG-2 恰恰迎合了数字 TV/HDTV 对压缩以及码率的要求,取代了 MPEG-3 这个最初为 HDTV 设计的标准。

MPEG-2 音频/视频压缩标准也称作 ISO/IEC13818[8],是由 ISO/IEC 的运动图像专家组在 1994 年 11 月提出的。与 MPEG-1 类似,它也有系统、视频、音频、一致性和软件部分,以及其他一些方面。除了广播数字电视,在陆地、卫星以及光缆网络方面, MPEG-2 也已经被广泛接受,并用于很多应用中,如交互式电视、DVD 等。

MPEG-2 为不同的应用 (如低延迟的视频会议、可伸缩视频、HDTV) 定义了七种规格: Simple (简单)、Main (主要)、SNR Scalable (SNR 可伸缩)、Spatially Scalable (空间可伸缩)、High (高)、4:2:2 和 Multiview (多视图) (两个视图就被称为立体视频)。在每种规格中,定义了四个等级。如表 11-5 所示,不是所有的规格都有四个等级。例如,简单规格就仅有一个主要等级;然而高规格就没有低等级。

319

表 11-5 MPEG-2 中的规格和等级

级	简单规格	主要规格	SNR 可伸缩规格	空间可伸缩规格	高规格	4:2:2 规格	多视图规格
High		*			*		
High 1440		*		*	*		
Main	*	*	*		*	*	*
Low		*	*				

表 11-6 列出了主要规格中的四个等级,主要规格有最大量的数据和目标应用。例如, High 级别支持的图像分辨率为  $1920 \times 1152$ ,最大帧率为 60fps,最大像素率为  $62.7 \times 10^6$  每秒,编码后最大数据率为 80Mbps。Low 级别以源输入格式 (SIF) 视频为目标,因此, MPEG-2 兼容 MPEG-1。Main 级别用于 CCIR 601 视频,而 High 1440 和 High 级别分别用于欧洲 HDTV 和北美洲 HDTV。

表 11-6 MPEG-2 中主要规格的 4 个等级

级	最大分辨率	最大 fps	最大像素/s	最大编码数据率 (Mbps)	应用
High	$1920 \times 1152$	60	$62.7 \times 10^6$	80	电影
High 1440	$1440 \times 1152$	60	$47.0 \times 10^6$	60	商业高清数字电视
Main	$720 \times 576$	30	$10.4 \times 10^6$	15	电视工作室
Low	$352 \times 288$	30	$3.0 \times 10^6$	4	商业影带

DVD 视频规范仅允许四种显示分辨率:  $720 \times 480$ 、 $704 \times 480$ 、 $352 \times 480$  和  $352 \times 240$ 。因此 DVD 视频标准仅使用 MPEG-2 Main 规格中 Main 级别和 Low 级别的限制形式。

#### 11.3.1 支持隔行扫描视频

MPEG-1 仅支持逐行扫描视频。由于 MPEG-2 被数字广播电视采用,所以它必须支持隔行扫描视频,因为这是数字广播电视和 HDTV 的一个特性。

前面提到过,在隔行扫描视频中,每帧由两个域组成,这两个域分别称为顶域(top-field)和底域(bottom-field)。在帧图中,两个域的所有扫描行交叉形成一帧,然后被分成大小为 $16\times 16$ 的宏块,并用运动补偿进行编码。另一方面,如果将每个域看作是一个独立的图片,则称为域图(field-picture)。如图 11-6a) 所示,每个帧图可分为两个场图。左侧帧图的 16 条扫描行被分为右侧的两个域图,每个域图各 8 条扫描线。

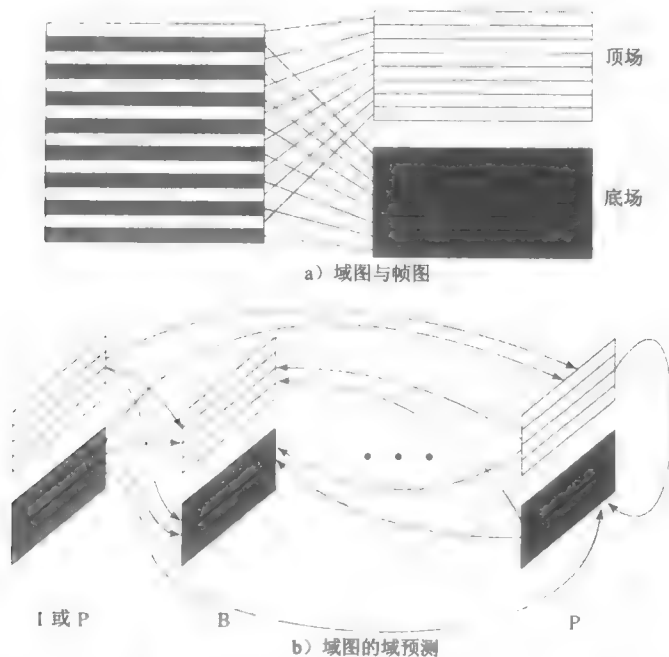


图 11-6 域图和 MPEG-2 中域图的场预测

我们可以看到,在显示器/TV 的显示区,域图中每个 $16\text{列}\times 16\text{行}$ 的宏块都对应于帧图中一个 $16\times 32$ 的块区域,而帧图中每个 $16\times 16$ 的宏块都对应于域图中一个 $16\times 8$ 的块区。下面的结论是为基于运动补偿的视频编码开发不同预测模式的重要因素。

### 1. 五种预测模式

MPEG-2 定义了帧预测(frame prediction)和域预测(field prediction)以及五种不同的预测模式,适用于对运动补偿的精度和速度有不同要求的许多应用。

#### (1) 帧图的帧预测

这种预测与 MPEG-1 中用于 P 帧和 B 帧的基于运动补偿的预测方法相同。帧预测仅适用于含有中慢速的物体和照相动作的视频。

#### (2) 域图的域预测

域图的域预测见图 11-6b)。这种模式用一个场图中大小为 $16\times 16$ 的宏块。对于 P 场图(图中最右侧的图)来说,由最近两次的编码域进行预测。顶域图中的宏块由先前的 I 帧或 P 帧的顶域或者底域图进行前向预测得到。底域图中的宏块由当前帧的顶域图或者先前 I 帧或 P 帧的底域图进行预测得到。

对于 B 场图,不论前向预测还是后向预测,都依照之前和之后的 I 帧或 P 帧的域图进行。没有规则要求维护域的“奇偶”,也就是说,顶域图和底域图既可以由参考图的顶域预测也可以由参考图的底域预测。

### (3) 帧图的域预测

这种模式将帧图的顶域和底域分开处理。相应地, 目标帧图的每个  $16 \times 16$  宏块可拆成分别来自两个域的两个  $16 \times 8$  部分。为这两个部分进行域预测的方式如图 11-6b 所示。除了块比较小外, 唯一不同就是底域不能从当前帧的顶域预测出来, 因为我们现在处理的是帧图。

例如, 对于 P 帧图, 底部  $16 \times 8$  部分将改为依照预测的 I 帧或 P 帧的域进行预测。对于 P 帧图的每个  $16 \times 16$  宏块将产生两个运动向量。同理, 对于 B 帧图的每个宏块将产生四个运动向量。

### (4) 域图的 $16 \times 8$ 运动补偿

目标域图的每个  $16 \times 16$  宏块分为顶部和底部两个  $16 \times 8$  的部分, 也就是前八行为一部分, 后八行为一部分。对于每个部分分别进行场预测。这样, 对于 P 场图中的每个  $16 \times 16$  宏块将产生两个运动向量, B 场图中的每个宏块将产生四个运动向量。这种模式对于速度快且无规律的运动可以进行很好的运动补偿。

### (5) P 图的对偶素

这是唯一一种既可以用于帧图又可以用于域图的模式。首先, 对具有相同奇偶性(顶或底)的每一个之前的域进行场预测。然后, 考虑到时间的伸缩以及顶场与底场行间的垂直移动, 每一个运动向量 MV 用来得到一个与之有相反奇偶性场的计算出的运动向量。这样, 这个 MV 与 CV 对为每个宏块产生两个初始预测。将它们的预测误差取平均作为最终的预测误差。这个模式的目的在于利用不使用后向预测的 P 图来模仿 B 图预测(因此减少了编码时延)。

## 2. 交替扫描和域离散余弦变换(Field-DCT)

交替扫描和域离散余弦变换(Field-DCT)技术的目的是提高 DCT 在预测误差上的有效性。这种技术只应用在隔行视频的帧图上。

在帧图上进行帧预测后, 预测误差将发送到每个块大小都是  $8 \times 8$  的 DCT。由于隔行视频的特性, 这些块中连续的行都来自不同的场; 因此它们之间的相关性比起隔行之间的相关性要小。这也就使低垂直空间频率的 DCT 系数的量级小于逐行视频的 DCT 系数。

基于之前的分析, 引入交替扫描。它可以应用于遵循 MPEG-2 的每张图, 而且是 MPEG-2 中 Z 字扫描的一个替代项。如图 11-7a 所示, 假设对一个逐行视频做 Z 字扫描, 块的左上角的 DCT 系数往往会有较大的量级。交替扫描(如图 11-7b 所示)在隔行视频中, 垂直较高空间频率分量会有较大的量级并可以在序列中先对它们进行扫描。实验表明[7], 交替扫描可以使峰值信噪比(PSNR)比 Z 字扫描改善近 0.3dB, 而且对于有快速运动的视频尤为有效。

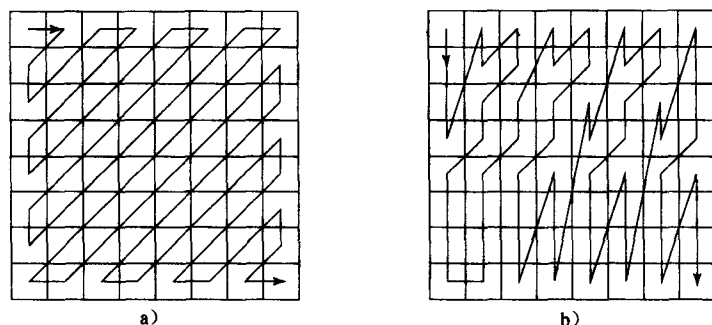


图 11-7 MPEG-2 中视频的 DCT 系数的 Z 形(逐行)扫描和交替(隔行)扫描

在 MPEG-2 中, 域 DCT 用于解决相同的问题。在应用 DCT 之前, 帧图宏块中的行被记录下来, 这样前八行来自顶域, 后八行来自底域。这就重复存储了连续行之间的较高空间的冗余(和

相关性)。这个记录在 IDCT 后还将被继续保留。域 DCT 不适用于每个宏块仅有  $8 \times 8$  像素的色度图像。

### 11.3.2 MPEG-2 的可伸缩性

和 JPEG2000 一样, 可伸缩性 (Scalability) 对于 MPEG-2 来说也是一个很关键的问题。因为 MPEG-2 适合多种应用, 包括数字电视和 HDTV, 视频往往以不同的形式在网络上传输。因此, 实现可变码率的单一编码位流是很有必要的。

MPEG-2 的可伸缩编码 (scalable coding) 也称为层次 (layered) 编码, 其中定义了一个基层和一个或多个增强层。基层可通过独立编码、传输和解码来获得基本的传输质量。增强层的编码和解码依赖于基层以及之前的增强层的。一般来说, 只使用一个增强层, 这种情况称为两层可伸缩编码。

可伸缩编码适合具有以下特征的 MPEG-2 视频在网络上的传输。

- **不同的码率** 如果连接速度较慢 (如使用 56kbps 的调制解调器拨号上网), 那么只发送基层的位流, 否则要发送一个或多个增强层的位流来提高视频质量。
- **可变码率 (VBR) 信道** 当信道的码率下降时, 将会少传 (或不传) 增强层的位流, 反之亦然。
- **低质量的连接** 基层可以被很好的保护或者通过一条质量较好的信道传送。

而且, 可伸缩编码也非常适用于逐行传输。首先发送基层的位流, 给用户一个快速而基本的视频视图, 接着逐步增加数据, 改进质量。这种方法很适合传输兼容的数字电视和 HDTV。

MPEG-2 支持如下的可伸缩性:

- **SNR 可伸缩性** 增强层提供较高的 SNR。
- **空间可伸缩性** 增强层提供较高的空间分辨率。
- **时间可伸缩性** 增强层有利于提供较高的帧率。
- **混合可伸缩性** 以上三种可伸缩性任意两种的组合。
- **数据划分** 拆分量化 DCT 系数。

#### 1. SNR 可伸缩性

图 11-8 描述了在 MPEG-2 编码器和解码器中 SNR 可伸缩性的工作原理。

MPEG-2 SNR 可伸缩编码器在两个层上产生输出位流 Bits\_base 和 Bits\_enhance。在基层对 DCT 系数进行较粗略的量化, 这样就产生较少的比特和质量较低的视频。经过变长编码后的位流称为 Bits\_base。

接下来, 对经过粗略量化的 DCT 系数进行逆量化 ( $Q^{-1}$ ) 并反馈给增强层, 与原来的 DCT 系数比较。对它们之间的差进行量化产生 DCT 系数修订, 并对其进行变长编码, 产生的位流称为 Bits\_enhance。将粗略的逆量化和精化后的 DCT 系数相加, 经过逆 DCT (IDCT), 用来对下一帧进行运动补偿预测。可见, 对基层进行增强/精化可以改进信噪比, 这种类型的可伸缩性称为 SNR 可伸缩性 (SNR scalability)。

如果由于某些原因 (如网络通道故障), 不能获取增强层的 Bits\_enhance, 那么上述可伸缩性方案仍然适用。在这种情况下, 增强层的逆量化器 ( $Q^{-1}$ ) 的输入为零。

解码器 (参见图 11-8b) 的操作正好与编码器的顺序相反。在将 Bits\_base 与 Bits\_enhance 相加恢复为 DCT 系数之前, 先对它们进行变长解码 (VLD) 和逆量化 ( $Q^{-1}$ )。接下来的步骤与基于运动补偿的视频解码器的处理步骤相同。如果两个位流 (Bits\_base 和 Bits\_enhance) 都被使用, 那么视频输出 (Output\_high) 将有很高的质量。如果仅仅用到了 Bits\_base, 那么输出视频 (Output\_base) 仅具有基本的质量。



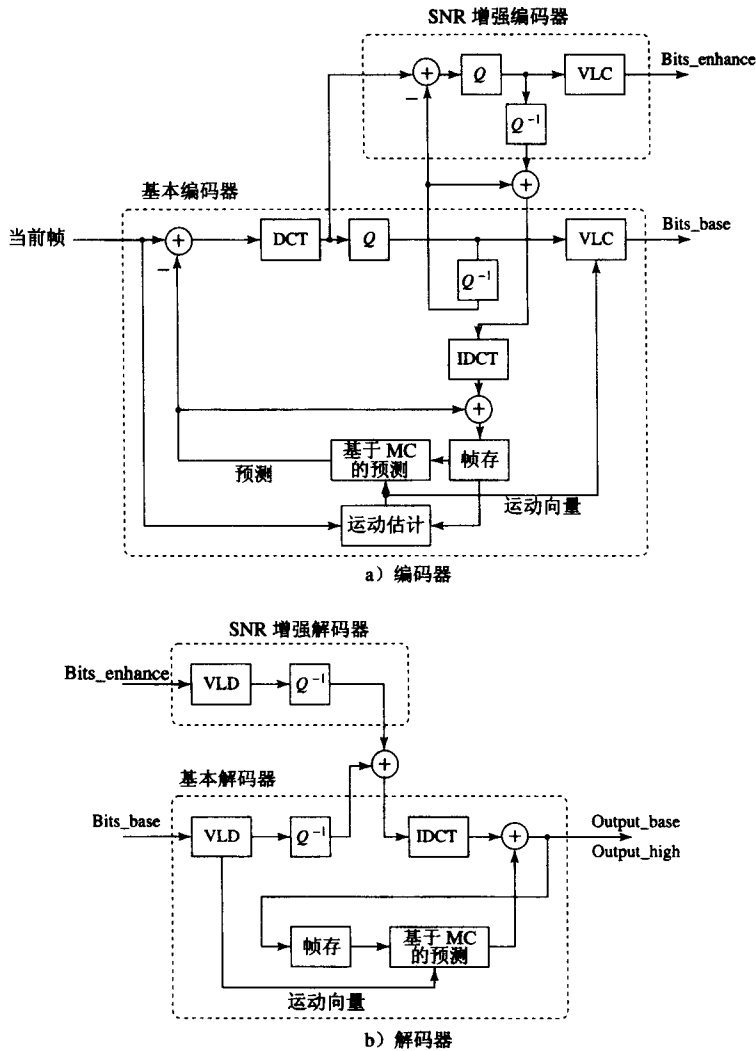


图 11-8 MPEG-2 SNR 可伸缩性

## 2. 空间可伸缩性

MPEG-2 空间可伸缩性的基层和增强层不像在 SNR 可伸缩性中联系得那么紧密，因此，这种可伸缩性比较简单。我们不会像上面那样对编码器和解码器进行详细介绍，我们利用高层次图表，仅对编码过程进行解释。

基层用来为低分辨率的图像产生位流。将它与增强层结合将产生初始分辨率的图像。如图 11-9a 所示，原始视频数据在空间上删减二分之一并发送到基层编码器。在通常的运动补偿编码、对预测误差进行 DCT、量化以及熵编码之后，输出位流为 Bits\_base。

如图 11-9b 所示，由基层产生的预测宏块经过空间插值得到  $16 \times 16$  的分辨率，接着将它与标准的、从增强层自身的时间预测宏块相结合，形成预测宏块，作为该层编码中的运动补偿。这里的空间插值采用前面讨论过的双线性插值技术。

利用一个简单的权值表来进行宏块之间的结合，其中权值  $w$  的范围限制在 0~1.0 之间。如果  $w=0$ ，表明对于从基层得来的预测宏块不加考虑。如果  $w=1$ ，则表明预测全部由基层得来的。通

常情况下,两个预测宏块分别利用加权  $w$  和  $1-w$  进行线性的结合。为了使预测误差最小,MPEG-2 的编码器内有一个从宏块权值表中选择不同  $w$  值的分析器。

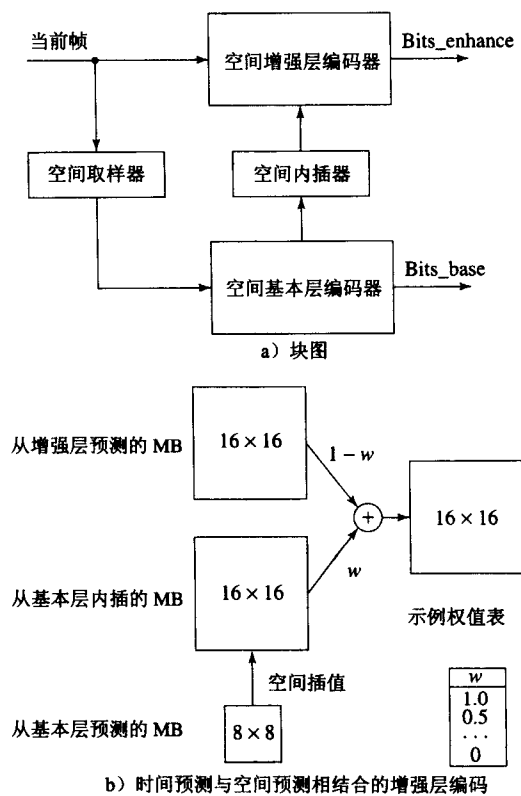


图 11-9 MPEG-2 空间可伸缩性编码器

3. 时间可伸缩性

减小帧率时时间可伸缩编码既有基层视频又有增强层视频。每层的减小帧率通常是相同的,但也可以不同。基层和增强层的图片与输入的视频有相同的空间分辨率。当整合到一起时,将视频重新恢复为初始的帧率。

图 11-10 描述了时间可伸缩性 MPEG-2 的实现。输入视频在时间上分成两部分,每部分的帧率是原来的一半。像之前描述的那样,基层编码器对输入它的视频执行普通的单层编码过程,并产生输出位流 Bits\_base。

对于增强层匹配宏块的预测可以采用下列两种方式[7]:隔层运动补偿预测或者运动补偿预测与隔层运动补偿预测相结合。

- **隔层运动补偿预测** 图 11-10b 解释了隔层运动补偿预测的原理。增强层运动补偿 B 帧的宏块由之前或者之后基层的帧预测得来(可以是 I、P 或是 B),这样可以在运动补偿中开发可能的层间冗余。
- **运动补偿预测与隔层运动补偿预测相结合** 图 11-10c 解释了这种方式。这种方式将常见的前向预测与上述的隔层预测的优点进一步结合。增强层 B 帧的宏块是由其本层的前向帧经前向预测以及基层的前向(或后向)帧经后向预测得来的。对于第一帧,增强层的 P 帧仅由基层的 I 帧经前向预测得到。

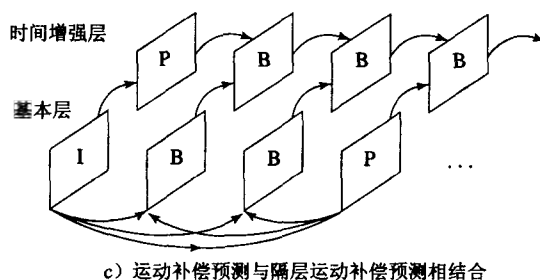
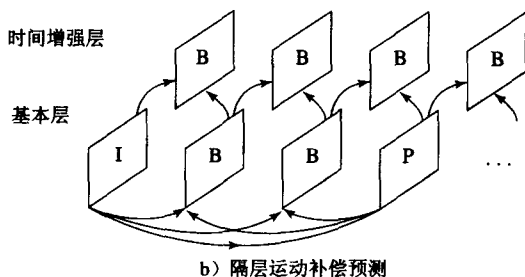
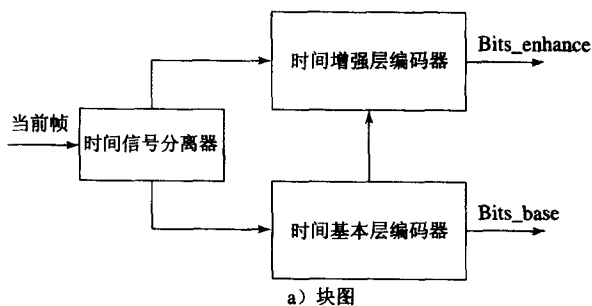


图 11-10 MPEG-2 时间可伸缩性编码器

#### 4. 混合可伸缩性

上面三种可伸缩性的任意两种组合即为混合可伸缩性。组合有以下几种：

- 空间时间混合可伸缩性
- SNR 空间混合可伸缩性
- SNR 时间混合可伸缩性

一般采用一个三层混合编码器，由基层、增强层 1 和增强层 2 组成。

以空间时间混合可伸缩性为例，基层和增强层 1 提供空间可伸缩性，而增强层 1 和增强层 2 提供时间可伸缩性，在这里增强层 1 可作为基层。

对于编码器，输入的视频数据首先在时间上分为两路数据流：一路进入增强层 2；另一路进入增强层 1 和基层（在对基层进行进一步空间抽取后）。

编码器产生三路输出位流：从基层来的  $\text{Bits\_base}$ ，从增强层 1 来的在空间上增强的  $\text{Bits\_enhance1}$ ，以及从增强层 2 来的在空间时间上增强的  $\text{Bits\_enhance2}$ 。

其他两种混合可伸缩性的实现与此类似，留做课后练习。

#### 5. 数据划分

压缩的视频流被划分为两个区，基础区包含低频 DCT 系数，增强区包含高频 DCT 系数。虽然有时区也称作层（基层和增强层），但严格来说，数据划分并不产生相同类型的层次编码，因

为一个视频数据流只是被划分,在产生增强区的时候并不依赖于基础区。不过,数据划分对于在嘈杂的信道上传输以及渐进传输是很有用的。

### 11.3.3 与 MPEG-1 的其他主要区别

- **更好地从位错误中恢复** 由于 MPEG-2 视频常常在不同的网络上进行传输,有些网络噪声很大而且不可靠,因此出现位错误是不可避免的。为了解决此问题, MPEG-2 系统有两种类型的流:程序流和传送流。程序流与 MPEG-1 中的系统流类似,因此,它可以很容易地兼容 MPEG-1。

传送流目的是进行差错恢复以及将具有不同扫描基线的多个程序合成一个单一的流来进行异步多路技术和网络传输。与 MPEG-1 所用的较长的变长包不同, MPEG-2 的程序流用的是定长包(188B)。而且有一个新的头语法,可以更好地进行差错检测和修复。

- **支持 4:2:2 和 4:4:4 的色度二次采样** 除了像 H.261 和 MPEG-1 那样支持 4:2:0 的色度二次采样, MPEG-2 还支持 4:2:2 和 4:4:4 的色度二次采样,以提高色彩质量。像第 5 章所讨论的那样,每个 4:2:2 的色度图片在水平方向除 2 进行二次采样, 4:4:4 是个特例,即没有进行色度二次采样。
- **非线性量化** MPEG-2 中的量化与 MPEG-1 中的量化相似。它的  $step\_size$  同样是由  $Q[i, j]$  与  $scale$  的乘积决定的,其中  $Q$  是帧内或帧间编码默认的量化表之一。允许的尺度类型有两种。一种是与 MPEG-1 相同的尺度,即[1,31]范围内的一个整数且  $scale_i = i$ 。另一种类型中存在一个非线性关系,即  $scale_i \neq i$ 。第  $i$  个尺度值可以在表 11-7 中查到。

表 11-7 MPEG-2 中可能的非线性尺度

$i$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$scale_i$	1	2	3	4	5	6	7	8	10	12	14	16	18	20	22	24
$i$	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
$scale_i$	28	32	36	40	44	48	52	56	64	72	80	88	96	104	112	

- **更严格的宏块片结构** MPEG-1 允许宏块片跨越宏块行边界。这样,一幅图片可以只是一个宏块片。而 MPEG-2 的宏块片必须开始结束于相同的宏块行。换句话说,一幅图片的左边界总是开始一个新的宏块片, MPEG-2 中最长的宏块片只能包含一个宏块行。
- **更灵活的视频格式** 依据标准, MPEG-2 图片的大小可以为 16K×16K 像素。在现实中, MPEG-2 主要用来支持 DVD、ATV 以及 HDTV 定义的不同的图片分辨率。

与 H.261、H.263 和 MPEG-1 类似, MPEG-2 仅指定了位流语法和解码器。这为将来的改进提供了很大空间,特别是编码器端的改进。MPEG-2 视频流语法比 MPEG-1 复杂得多,具体信息可以参考文献[8, 7]。

## 11.4 进一步探索

Mitchell et al.[6]和 Haskell et al.[7]的书籍对 MPEG-1 和 MPEG-2 进行了详细介绍。Haskell et al.的文章[9]概述了数字视频编码标准。

本书网站上本章的 Further Exploration 部分给出了各类资源的 URL,包括 MPEG 的主页、FAQ 页、MPEG-1 和 MPEG-2 标准的综述以及工作文档。

## 11.5 练习

1. 众所周知, MPEG 视频压缩利用了 I 帧、P 帧和 B 帧。然而,早期的 H.261 标准并没有用到 B

帧。描述一种由于没有使用 B 帧而使视频压缩性能欠佳的情况。(答案不得与图 11-1 中的情况相同。)

2. 默认的帧间量化表  $Q_2$  都是常数, 而帧内量化表  $Q_1$  则不是常数, 解释原因。
3. 与 MPEG-1 相比, MPEG-2 有哪些增强? 为什么 MPEG-2 标准没有替代 MPEG-1 标准?
4. B 帧给编码带来了很大的好处, 如在低码率的情况下增加了 SNR 而且节省带宽, 那么 B 帧有哪些缺点?
5. MPEG-1 标准引入了 B 帧, 使运动向量搜索范围也相应地由 H.261 中的  $[-15, 15]$  增加到  $[-512, 511.5]$ 。为什么这是必然的? 计算连续 P 帧间的 B 帧个数可以证明搜索范围的增长。
6. 重新绘制图 11-8 中 MPEG-2 两层 SNR 可伸缩性编码器和解码器, 使之包含第二个增强层。
7. 绘制下列可伸缩性的 MPEG-2 编码器和解码器的块图: (a) SNR 空间混合可伸缩性, (b) SNR 时间混合可伸缩性。
8. 为什么 B 帧不能在运动补偿中作为参考帧? 假设任何类型的帧都可作为参考帧。讨论在视频序列中用参考 B 帧代替 P 帧的利弊(即完全消除 P 帧)。
9. 设计一种在一个 SNR 可伸缩性 MPEG-2 位流的增强层使用运动补偿的方法。为什么在 MPEG-2 的标准中没有这样建议?
10. 写一段可以在 MPEG-2 中实现 SNR 可伸缩性的程序。该程序应能运行在任何宏块上、使用任意量化步长  $step\_sizes$ , 并能输出 Bits\_base 和 Bits\_enhance 位流。不要忽略可变长编码这个步骤。

## 11.6 参考文献

- [1] L. Chiariglione, "The Development of an Integrated Audiovisual Coding Standard: MPEG," *Proceedings of the IEEE*, 83: 151–157, 1995.
- [2] L. Chiariglione, "Impact of MPEG Standards on Multimedia Industry," *Proceedings of the IEEE*, 86(6): 1222–1227, 1998.
- [3] R. Schafer and T. Sikora, "Digital Video Coding Standards and Their Role in Video Communications," *Proceedings of the IEEE*, 83(6): 907–924, 1995.
- [4] D.J. LeGall, "MPEG: A Video Compression Standard for Multimedia Applications," *Communications of the ACM*, 34(4): 46–58, 1991.
- [5] *Information Technology—Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s*, International Standard: ISO/IEC 11172, Parts 1–5, 1992.
- [6] J.L. Mitchell, W.B. Pennebaker, C.E. Fogg, and D.J. LeGall, *MPEG Video Compression Standard*, New York: Chapman & Hall, 1996.
- [7] B.G. Haskell, A. Puri, and A. Netravali, *Digital Video: An Introduction to MPEG-2*, New York: Chapman & Hall, 1997.
- [8] *Information Technology—Generic Coding of Moving Pictures and Associated Audio Information*, International Standard: ISO/IEC 13818, Parts 1–10, 1994.
- [9] B.G. Haskell, et al., "Image and Video Coding: Emerging Standards and Beyond," *IEEE Transactions on Circuits and Systems for Video Technology*, 8(7): 814–837, 1998.

# 第 12 章 MPEG 视频编码 II：MPEG-4、MPEG-7 及更高版本

## 12.1 MPEG-4 概述

MPEG-1 和 MPEG-2 使用了基于帧的编码技术，其中每个矩形视频帧作为一个压缩单元，这种压缩技术主要关注高压缩率和令人满意的图像质量。MPEG-4 是一个更新的标准[1]，除了压缩外，该标准还关注用户交互，这就使大量的用户能够通过新的基础设施（如因特网、WWW 以及移动/无线网络）创建和交流他们的多媒体演示和应用。MPEG-4 与之前版本的不同之处在于它采用了一种新的基于对象编码（object-based coding）的方式，此时媒体对象成为 MPEG-4 的编码实体。媒体对象（也称为音频和视频对象）可以是自然的，也可以是合成的，也就是说，它们可以由摄像机采集的，也可以是由计算机程序生成的。

基于对象编码不仅可以提供更高的压缩率，而且利于进行数字视频合成、处理、索引和检索。图 12-1 说明如何通过对可视对象进行插入/删除、平移/旋转、缩放等简单操作，对 MPEG-4 视频进行合成和处理。

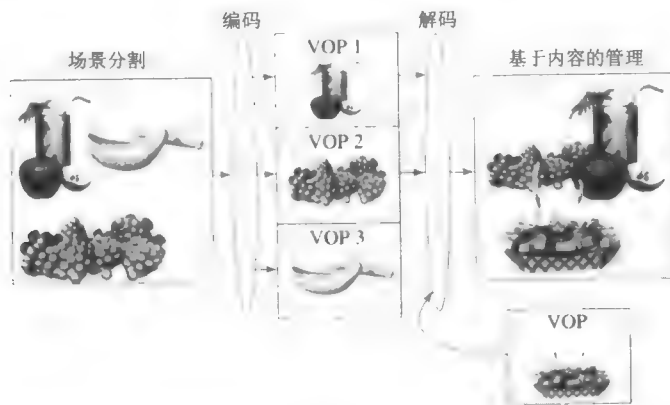


图 12-1 MPEG-4 视频的合成和处理（VOP = Video Object Plane，即视频对象平面）

MPEG-4(版本 1)完成于 1998 年 10 月，并于 1999 年初成为国际标准，称为 ISO/IEC 14496[2]；改进版本（版本 2）完成于 1999 年 12 月，并于 2000 年成为国际标准。与前一个 MPEG 标准相似，它的前 5 个部分是系统、视频、音频、一致性和软件。它的第 6 个部分，即多媒体传送整体框架（Delivery Multimedia Integration Framework, DMIF）是新的，我们将在第 16 章对其进行深入讨论，在该章中我们将讨论多媒体网络通信和应用。

MPEG-4 视频的最初适用于低码率的通信（对移动通信是 4.8~64kbps，对其他应用最高到 2Mbps），现在它的码率已经扩大到 5kbps~10Mbps。

如图 12-2a 中的参考模型所示，MPEG-1 系统仅仅从它的存储区传送音频和视频数据，没有用户交互。MPEG-2 增加了交互组件（在图 12-2a 中用虚线表示），因此可以提供如网络视频和交互电视等有限的用户交互。MPEG-4（如图 12-2b 所示）是一个全新的标准，可以合成媒体对象以创建所需的视听场景以及多路传播和同步媒体数据项的位流，以保证它们传输过程中的服务质量

(Quality of Service, QoS), 并能在接收端与视听场景进行交互。MPEG-4 为音频和视频压缩提供了高级编码模块和算法工具箱。

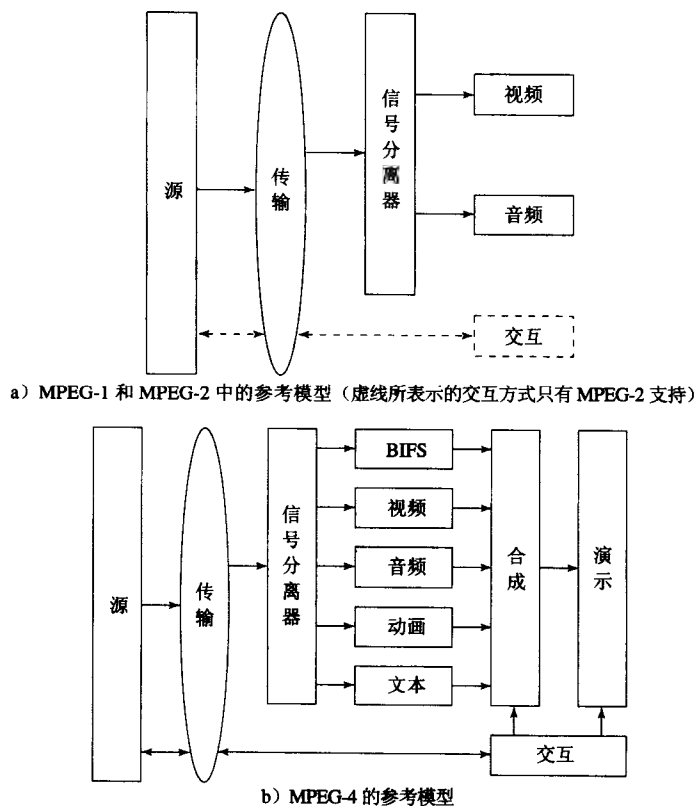


图 12-2 MPEG 标准中的交互的对比

MPEG-4 定义了场景的二进制格式 (Binary Format for Scenes, BIFS) [3], 以便于将媒体对象合成到场景中。BIFS 通常用一个场景图表表示, 其中节点描述了视听对象及其属性, 而图结构可以给出场景中各对象空间和时间关系的描述。BIFS 是对虚拟现实建模语言 (VRML) 的增强。实际上, 它强调了对对象的时序和同步, 这是最初的 VRML 设计所缺乏的。除了 BIFS 之外, MPEG-4 (版本 2) 还提供了一个编程环境 MPEG-J[4], 其中 Java 应用程序 (称为 MPEGlets) 可以访问 Java 包和 API 以增强终端用户的交互。

MPEG-4 的视觉位流的层次结构与 MPEG-1 和 MPEG-2 的层次结构有很大的不同, 它是面向视频对象的。图 12-3 给出了 MPEG-4 视觉位流中一个场景的层次结构描述中的 5 个层次。通常, 每一个视频对象序列 (Video-object Sequence, VS) 都拥有一个或多个视频对象 (Video Object, VO), 每一个 VO 都拥有一个或多个视频对象层 (Video Object Layer, VOL)。按照语法, 5 个层次在位流中都有唯一的开始码, 以便能够随机访问。

### 1. 视频对象序列

VS 传送完整的 MPEG-4 视觉场景, 可以包含 2D 或 3D 的自然对象或合成对象。

### 2. 视频对象

VO 是场景中的一个特殊对象, 它可以拥有任意的 (非矩形的) 形状, 与场景中的一个对象或背景相对应。

332  
}  
334

3. 视频对象层

VOL 提供了一种便捷的方式支持（多层的）可扩展的编码。一个 VO 在可扩展编码下可以拥有多重 VOL，在非扩展编码下拥有一个 VOL。作为一个特例，MPEG-4 也支持一种特殊的 VOL，它的头部较短，这能使位流与 H.263[5]基本系统兼容。

4. 视频对象平面组

GOV 将视频对象平面分组，是一个可选的层。

5. 视频对象平面

一个 VOP 是一个 VO 在特定时刻的快照，反映了该时刻 VO 的形状、纹理和运动参数。一般来说，一个 VOP 是一个任意形状的图像。当把整个矩形的视频帧当作一个 VOP 的时候，MPEG-4 视频编码中会出现一种退化的情形。在这种情况下，它等同于 MPEG-1 和 MPEG-2。MPEG-4 允许重叠的 VOP，也就是说，在一个场景中一个 VOP 可以部分覆盖另一个 VOP。

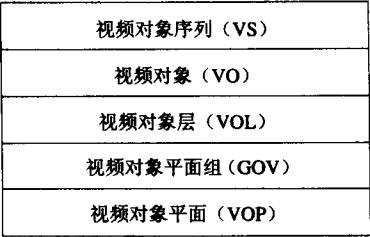


图 12-3 MPEG-4 可视位流中场景的面向视频对象的分层描述

12.2 MPEG-4 的基于对象的视觉编码

MPEG-4 分别编码/解码每个 VOP（而不是考虑整个帧）。因此，它的基于对象的视觉编码也称为基于 VOP 的编码。我们的讨论从自然对象（可以在[6][7]中查找到更多细节）的编码开始，12.3 节将介绍合成对象编码。

12.2.1 基于 VOP 的编码与基于帧的编码

335

MPEG-1 和 MPEG-2 不支持 VOP 的概念，因此它们的编码方式称为基于帧的。因为每一帧都被分为许多宏块，利用这些宏块进行基于运动补偿的编码，所以这种方法也称为基于块的编码（block-based coding）。图 12-4a 显示了一段视频序列中的 3 个帧，其中一辆汽车向左运动，而一个步行者朝反方向走。图 12-4b 显示了典型的基于块的编码，其中运动向量（MV）由其中一个宏块获得。

MPEG-1 和 MPEG-2 视觉编码只关心压缩率，并不考虑视觉对象的存在。因此，生成的运动向量可能与对象的运动不一致，所以不利于基于对象的视频分析和索引。

图 12-4c 说明了一个可能的例子，其中两个可能的匹配都生成小的预测误差。如果可能匹配 2 生成的预测误差比可能匹配 1 生成的预测误差小，那么在基于块的编码方式中 MV2 将被选为宏块的运动向量，虽然只有 MV1 与汽车的运动方向一致。

MPEG-4 中基于对象的编码方式除了改进压缩之外，还可以解决上述问题。图 12-4d 显示了每个 VOP 可以拥有任意的形状，并且在理想情况可以获取一个唯一的与对象运动一致的运动向量。



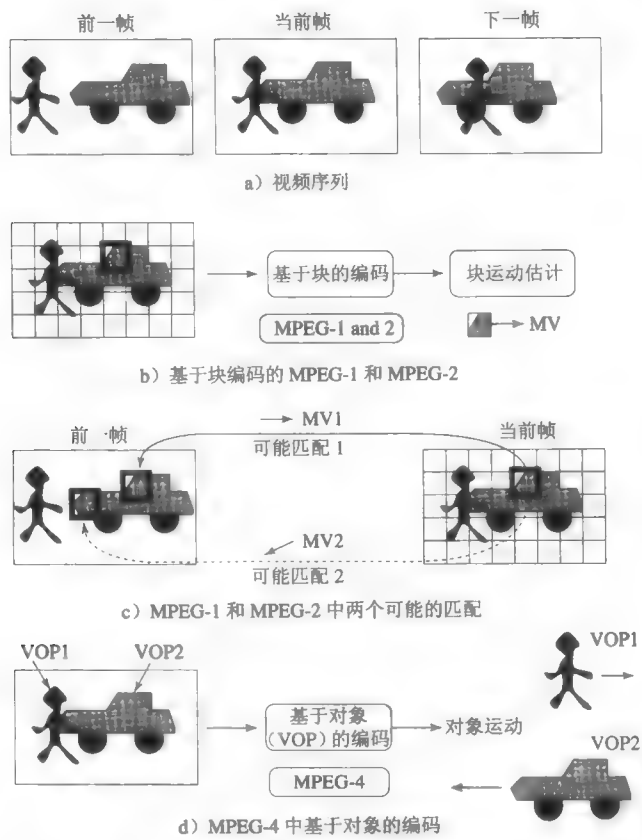


图 12-4 基于块的编码方式和基于对象的编码方式的对比

MPEG-4 基于 VOP 的编码方式也使用运动补偿技术。一个帧内编码的 VOP 又称为一个 I-VOP，只使用前向预测的帧间编码的 VOP 被称为 P-VOP，而使用双向预测的帧间编码 VOP 则被称为 B-VOP。此时，新的难点在于 VOP 可能拥有任意的形状。因此，除了它们的纹理，它们的形状信息也必须进行编码。

注意，这里的纹理实际上指视觉内容，也就是 VOP 中各像素的灰度和色度值。MPEG-1 和 MPEG-2 不会对形状信息编码，因为所有的帧都是矩形，但会对帧中的像素值编码。在 MPEG-1 和 MPEG-2 中，这种编码并没有被称为纹理编码。“纹理”这个术语来自计算机图形学，它表明这门学科与 MPEG-4 一道进入了视频编码的领域。

下面我们从讨论 VOP 的基于运动补偿的编码开始，接着介绍纹理编码、形状编码、静态纹理编码，子图像 (Sprite) 编码以及全局运动补偿。

### 12.2.2 运动补偿

本节将阐述 MPEG-4 中基于 VOP 的运动补偿的问题。因为 I-VOP 编码相对简单，所以除非明确提到 I-VOP，否则我们讨论的都是 P-VOP 和 B-VOP 的编码。

与以前一样，MPEG-4 中的基于运动补偿的 VOP 编码涉及三个步骤：运动估计、基于运动补偿的预测和预测误差的编码。为便于运动补偿，每一个 VOP 都被分为许多宏块，这和前面介绍的基于帧的编码方式一样。宏块默认的是 16×16 的亮度图和 8×8 的色度图，当它们跨越一个不规则形状的 VOP 的边界时会有特别处理。

MPEG-4 为每一个 VOP 定义了一个矩形的边界框，其左边界和上边界是 VOP 的左边界和上边界，确定了 VOP 从用于视频帧的绝对坐标系下原点 (0,0) 处平移后的原点位置（如图 12-5 所示）。亮度图中边界框在水平和垂直方向上都必须是 16 的倍数，因此，此框通常比常规的边界框要大一些。

完全处于 VOP 内部的宏块称为内部宏块 (interior macroblock)。从图 12-5 中可以明显地看出，许多宏块跨越了 VOP 的边界，这些宏块称为边界宏块 (boundary macroblock)。

内部宏块的运动补偿方式与 MPEG-1 和 MPEG-2 的运动补偿方式相同。但是边界宏块在运动估计中将很难匹配，因为 VOP 通常形状不规则，而且它们的形状在视频中的不同时刻可能会发生变化。为便于匹配目标 VOP 的每个像素并且满足变换编码（例如 DCT）中对矩形块的强制要求，在运动估计前，将对参考 VOP 应用称为填充 (padding) 的预处理步骤。

只有当前（目标）VOP 内部的像素才需要进行运动补偿的匹配，而填充仅仅针对参考 VOP。为得到更好的质量，当然还可以开发一些比填充更好的扩展方法，在 MPEG-4 中采用填充方法，很大程度上是因为该方法简单、高速。

运动补偿的前两个步骤是填充和运动向量编码。

1. 填充

对于参考 VOP 中所有的边界宏块，首先进行水平重复填充 (horizontal repetitive padding)，接着进行垂直重复填充 (vertical repetitive padding)（如图 12-6 所示）。在此之后，对所有 VOP 之外、但与一个或多个边界宏块相邻的外部宏块 (exterior macroblock) 进行扩展填充 (extended padding)。

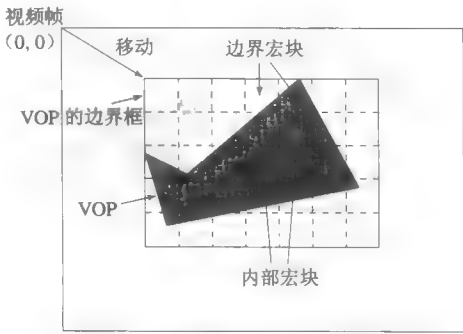


图 12-5 VOP 的边界框和边界宏块



图 12-6 MPEG-4 中参考 VOP 的填充顺序

水平重复填充算法扫描参考 VOP 中边界宏块的每一行，每一个边界像素将被向左或向右复制以填充宏块中 VOP 之外的中间像素值。如果这一段像素在两个边界像素之间，则采用这两个边界像素的平均值。

算法 12-1 水平重复填充

```
begin
  for all rows in Boundary macroblocks in the Reference VOP
    if  $\exists$  (boundary pixel) in the row
      for all interval outside of VOP
        if interval is bounded by only one boundary pixel  $b$ 
          assign the value of  $b$  to all pixels in interval
        else // interval is bounded by two boundary pixels  $b_1$  and  $b_2$ 
          assign the value of  $(b_1 + b_2)/2$  to all pixels in interval
end
```

接下来的垂直重复填充算法与水平重复填充算法相似。它扫描每一列，在水平填充过程中新

填充的像素被当作 VOP 内部的像素进行垂直填充。

### 例 12-1

图 12-7 解释了一个参考 VOP 的边界宏块中的重复填充的例子。图 12-7a 显示的是 VOP 中像素的亮度值（或色度值），VOP 的边界用加黑的线显示。为简单起见，此例中宏块的分辨率减至 6×6，尽管其实际的宏块大小是 16×16（亮度图）和 8×8（色度图）。

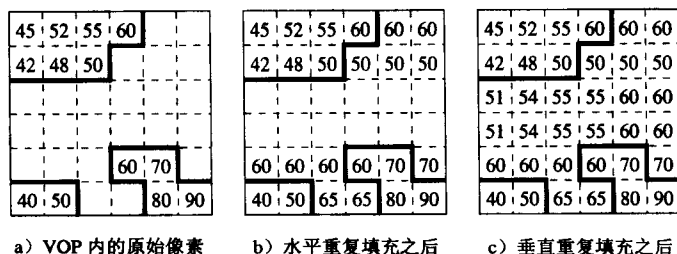


图 12-7 一个参考 VOP 边界宏块重复填充的例子

#### (1) 水平重复填充（参见图 12-7b）

- 第 0 行 VOP 的最右边的像素就是边界像素，其亮度值是 60，重复用作 VOP 外的像素值。
- 第 1 行 同样，VOP 的最右边的像素也是边界像素，其亮度值是 50，重复用作 VOP 外的像素值。
- 第 2 行和第 3 行 无水平填充，因为没有边界像素。
- 第 4 行 VOP 外有两个间隔，每个间隔都是由一个边界像素界定的，它们的亮度值是 60 和 70，分别重复用作两个间隔的像素值。
- 第 5 行 VOP 外的一个间隔由 VOP 的一对边界像素界定。其平均亮度值是  $(50+80)/2=65$ ，因此将 65 重复用作两者之间像素的值。

#### (2) 垂直重复填充（参见图 12-7c）

第 0 列 由 VOP 的一对边界像素界定的一个间隔。这两个边界像素分别是 42 和 60（是由水平填充产生的），它们的平均亮度是  $(42+60)/2=51$ ，将 51 重复用作两者之间的像素的值。

第 1~5 列的填充过程和第 0 列类似。

### 2. 扩展填充

全部位于 VOP 之外的宏块是外部宏块。紧邻边界宏块的外部宏块用边界宏块的边界像素值填充。我们知道，边界宏块现在已经完全被填充了，因此它们的水平和垂直边界像素都已经赋值。如果一个外部宏块拥有多个紧邻的边界宏块，那么按照左、上、右、下的顺序为其选择边界宏块进行扩展填充。

MPEG-4 的较新的版本允许利用这些边界宏块的平均值。扩展填充的过程可以重复应用到该 VOP 矩形边界框的所有外部宏块上。

### 3. 运动向量编码

目标 VOP 上的每一个宏块将通过如下的运动估计过程找到参考 VOP 的一个最佳匹配宏块。

设  $C(x+k, y+l)$  为目标 VOP 宏块中的像素，而  $R(x+i+k, y+j+l)$  为参考 VOP 宏块中的像素。

与式 10.1 中的 MAD 相似，为测量两个宏块的不同，定义如下所示的 SAD：

$$SAD(i, j) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} |C(x+k, y+l) - R(x+i+k, y+j+l)| \cdot Map(x+k, y+l)$$

其中,  $N$  表示宏块的大小。当  $C(p,q)$  是目标 VOP 中的一个像素时,  $Map(p,q)=1$ , 否则  $Map(p,q)=0$ 。产生最小 SAD 值的向量  $(i,j)$  作为运动向量  $MV(u,v)$ :

$$(u,v)=\{(i,j)|SAD(i,j) \text{ is minimum}, i \in [-p,p], j \in [-p,p]\} \quad (12.1)$$

其中  $p$  是  $u, v$  可以取的最大值。

对于运动补偿, 要给运动向量  $MV$  编码。如 H.263 (参见图 10-11) 所述, 并不是简单的取目标宏块的运动向量为  $MV$ 。 $MV$  是通过 3 个相邻的宏块预测得到的。然后对预测误差进行可变长编码。

以下是一些高级的运动补偿技术, 它们与 H.263 中采用的技术相似 (参见 10.5 节)。

- 可以为 VOP 的亮度分量中的每一个宏块生成 4 个运动向量 (每一个向量来自一个  $8 \times 8$  的块)。
- 运动向量可以进行比像素级更精确的预测。在半像素预测中, 运动向量的范围是  $[-2048, 2047]$ 。MPEG-4 还允许在 VOP 亮度分量上进行  $1/4$  像素预测。
- 可以使用不受限制的运动向量:  $MV$  可以指向参考 VOP 边界外。当参考 VOP 之外的像素时, 它的值仍是根据填充进行定义的。

### 12.2.3 纹理编码

纹理指 VOP 中的灰度级 (或色度) 的变化及样式。MPEG-4 中的纹理编码可以基于 DCT 或是形状自适应 DCT (shape Adaptive-DCT, SA-DCT)。

#### 1. 基于 DCT 的纹理编码

在 I-VOP 中, VOP 的各宏块像素的灰度 (或色度) 值先使用 VLC 接着使用 DCT 直接进行编码, 这与 JPEG 对静态图片的编码很相似。P-VOP 和 B-VOP 使用基于运动补偿的编码, 因此预测误差将传送 DCT 和 VLC。下面的讨论重点关注对 P-VOP 和 B-VOP 的基于运动补偿的纹理编码。

对内部宏块的编码与 H.261、H.263 及 MPEG-1 和 MPEG-2 中传统的基于运动补偿的编码相似 (其中亮度 VOP 为  $16 \times 16$  的块, 而色度 VOP 为  $8 \times 8$  的块)。在进行常规的运动估计后, 可获取每个宏块中 6 个  $8 \times 8$  的块的预测误差。这些误差将传送到 DCT 程序以获取 6 个  $8 \times 8$  的块的 DCT 系数。

对边界宏块来说, 参考 VOP 之外的区域通过重复填充过程进行填充, 这个过程前面已经介绍过。在运动补偿后, 获取目标 VOP 内的纹理预测误差。对于目标 VOP 边界宏块中 VOP 外的部分用 0 填充, 然后传给 DCT, 因为在理想情况下, VOP 内部的预测误差是接近 0 的。重复填充和扩展填充都是为了在运动补偿中有更好的匹配, 填充 0 则是为了在纹理编码中得到更好的 DCT 结果。

DC 分量的量化步长  $step\_size$  为 8。对于 AC 系数, 可以使用如下两种方法中的一种:

- H.263 方法, 其中所有的系数获得由一个参数控制的量化器, 不同的宏块拥有不同的量化器。
- MPEG-2 方法, 其中同一个宏块中的 DCT 系数可以有不同量化器, 由  $step\_size$  参数进一步控制。

#### 2. 边界宏块的 SA-DCT 编码

SA-DCT[8]是另一种对边界宏块进行纹理编码的方法。由于该方法效率较高, 因此在 MPEG-4 版本 2 中已经采用 SA-DCT 进行边界宏块的编码。

1D DCT-N 是前述的 1D DCT (见式 (8.19) 和式 (8.20)) 的变体, 其中变换使用可变量  $N$  代替固定的  $N=8$ 。(为简单起见, 本节中用 DCT-N 表示 1D DCT-N。)

式 12.2 和 12.3 描述了 DCT-N 变换及其逆变换 IDCT-N。

一维离散余弦变换-N (DCT-N)

$$F(u) = \sqrt{\frac{2}{N}} C(u) \sum_{i=0}^{N-1} \cos \frac{(2i+1)u\pi}{2N} f(i) \quad (12.2)$$

一维逆离散余弦变换-N (IDCT-N)

$$\tilde{f}(i) = \sum_{u=0}^{N-1} \sqrt{\frac{2}{N}} C(u) \cos \frac{(2i+1)u\pi}{2N} F(u) \quad (12.3)$$

其中  $i=0,1,\dots,N-1$ ;  $u=0,1,\dots,N-1$ , 并且

$$C(u) = \begin{cases} \frac{\sqrt{2}}{2}, & u=0 \\ 1, & \text{其他} \end{cases}$$

SA-DCT 是一个 2D DCT, 并作为可分的 2D 变换通过两次 DCT-N 迭代进行计算。图 12-8 说明了使用 SA-DCT 对边界宏块进行纹理编码的过程。变换应用于边界宏块的每个  $8 \times 8$  的块。

图 12-8a 显示了边界宏块中的一个  $8 \times 8$  块, 其中宏块中的像素用  $f(x, y)$  表示, 且显示为灰色。灰色像素首先被向上平移得到  $f'(x, y)$ , 如图 12-8b 所示。在第一次迭代中, DCT-N 应用到  $f'(x, y)$  的第一列, 其中每一列中灰色像素数决定了 N 的大小。因此, 我们分别使用 DCT-2、DCT-3、DCT-5 等。计算出的 DCT-N 系数由  $F'(x, v)$  表示, 如图 12-8c 所示, 其中黑色的点表示 DCT-N 的 DC 系数。 $F'(x, v)$  接着向左平移, 得到图 12-8d 中的  $F''(x, v)$ 。

在第二次迭代中, DCT-N 应用于  $F''(x, v)$  每一行, 得到  $G(u, v)$  (如图 12-8e 所示), 图中黑色的点表示 2D SA-DCT 的 DC 系数  $G(0, 0)$ 。

342

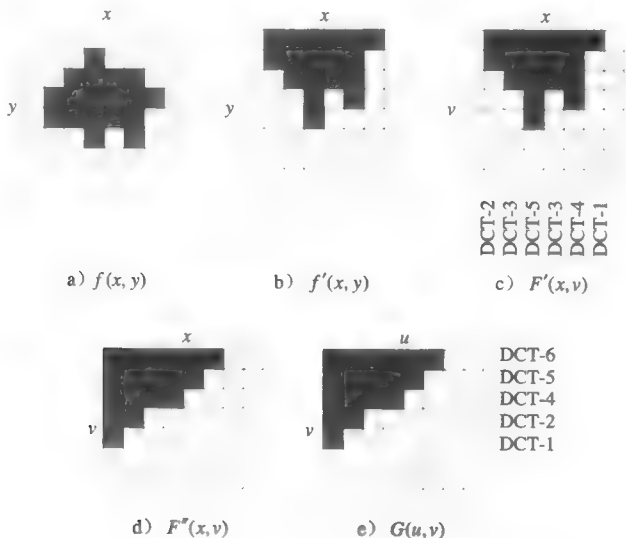


图 12-8 使用 SA-DCT 对边界宏块进行纹理编码

以下是一些编码时需要考虑的问题:

- $G(u, v)$  中 DCT 系数的总数等于边界宏块的  $8 \times 8$  块中的灰色像素数, 少于  $8 \times 8$ 。因此, 该

方法是形状自适应的, 并且计算效率更高。

- 在解码时, 因为在每次 IDCT-N 迭代后数组元素都需要适当地移动回来, 要求使用初始形状的二元掩码对用 SA-DCT 进行纹理编码的信息解码。二元掩码与下面将介绍的二元 $\alpha$ 图相同。

## 12.2.4 形状编码

与 MPEG-1 和 MPEG-2 不一样, MPEG-4 必须对 VOP 的形状进行编码, 因为形状是视觉对象一个固有的特性。

MPEG-4 支持两种形状信息: 二值和灰度。二值形状信息可以用二值图的形式 (也被称为二元 $\alpha$ 图) 表示, 这种图与 VOP 的矩形边界框大小相同。位图信息中值 1 (不透明) 和值 0 (透明) 分别表示像素是否在 VOP 内。灰度形状信息事实上是指形状的透明度, 灰度值范围为 0~255。

### 1. 二元形状编码

为了更高效地对二元 $\alpha$ 图编码, 将图分为  $16 \times 16$  大小的块, 这种块也称为二元 $\alpha$ 块 (Binary Alpha Block, BAB)。如果一个 BAB 完全透明或不透明的, 那么很容易编码, 不需要特别的形状编码技术。问题在于那些包含轮廓即 VOP 形状信息的边界 BAB。它们是二元形状编码的目标。

为对边界 BAB 进行编码[9], 研究人员研究和比较了各种基于轮廓和基于位图 (或者基于区域) 的算法, 最后选出的两种算法都是基于位图的。一种是 MMR (Modified Modified READ, 修订修改读) 算法, 它是传真 G3 标准[10]中可选的增强项及 G4 标准[11]中强制要求的压缩方式。另一种是基于上下文的算术编码 (Context-based Arithmetic Encoding, CAE), 它最初是为 JBIG[12]开发的, 由于其简单且压缩率高, CAE 最终被选为 MPEG-4 的二元形状编码方案。

MMR 基本上是 READ (Relative Element Address Designate) 算法的一系列简化。READ 算法的基本思想是根据前一个已编码行的像素位置对当前行进行编码。该算法先确定上一行和当前行中的 5 个像素点的位置:

- $a_0$ : 编码器和解码器所知的最后一个像素值。
- $a_1$ :  $a_0$  右边的过渡像素。
- $a_2$ :  $a_0$  右边的第二个过渡像素。
- $b_1$ : 上一个已编码行中第一个与  $a_0$  颜色相反的过渡像素。
- $b_2$ : 上一个已编码行中  $b_1$  右边的第一个过渡像素。

READ 算法通过扫描这些像素的相对位置进行工作。在任一时刻, 编码器和解码器都知道  $a_0$ ,  $b_1$  和  $b_2$  的位置, 而  $a_1$  和  $a_2$  的位置只有编码器知道。

该算法使用了三种编码模式:

- 如果前一行和当前行的游长相似, 那么  $a_1$  和  $b_1$  之间的距离应该远远小于  $a_0$  和  $a_1$  之间的距离。因此, 垂直模式 (vertical mode) 将当前的游长编码为  $a_1 - b_1$ 。
- 如果前一行和当前行的游长不相似, 当前行的游长使用一维游长编码。这种模式称为水平模式 (horizontal mode)。
- 如果  $a_0 \leq b_1 < b_2 < a_1$ , 我们可以只传送一个码字, 表示采用通道模式 (pass mode), 将  $a_0$  提前到  $b_2$  下面的位置继续并进行编码过程。

对 READ 算法的实际实现可以进行一些简化。例如, 如果  $\|a_1 - b_1\| < 3$ , 说明我们应该使用垂直模式。同时, 为避免误差传播, 定义一个  $k$  因子, 这样每  $k$  行中必须至少有一行使用传统的游长编码。这些修改构成了 G3 标准中的修订 READ 算法。MMR 算法只是去掉了  $k$  因子强加的限制。

对基于上下文的算术编码, 图 12-9 说明了边界 BAB 中一个像素的“上下文”。在 CAE 内模

式下,当仅涉及目标 $\alpha$ 图(图 12-9a))时,相同 $\alpha$ 图下的 10 个邻近像素(编号 0~9)就形成了上下文。与这些像素相关的 10 个二进制数可以提供至多  $2^{10}=1024$  种可能的上下文。

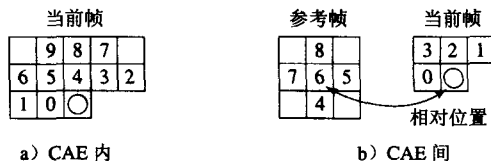


图 12-9 MPEG-4 中进行二元形状编码的 CAE 中的上下文。○表示当前的像素,数字表示相邻的其他像素

很显然,某些上下文(比如全为 1 或 0)出现的频率大于其他上下文。利用一些预先的统计,可以建立一个概率表表示这 1024 种上下文的出现频度。

回忆一下,算术编码(第 7 章中介绍的)可以用一个数字编码概率符号的顺序。现在,每个像素可以查找这个表以获取其上下文的概率值。CAE 仅仅在 BAB 中顺序地扫描每个  $16 \times 16$  像素并应用算术编码为该 BAB 计算出一个浮点数。

CAE 间模式是 CAE 内模式的自然扩展,它涉及目标 $\alpha$ 图和参考 $\alpha$ 图。对目标帧中的每一个边界宏块,首先调用运动估计(整数估计)和压缩过程找到参考帧中的匹配宏块。这确立了边界 BAB 中每个像素相应的位置。

图 12-9b 显示每个像素的上下文包括目标 $\alpha$ 图中 4 个相邻像素及参考 $\alpha$ 图中的 5 个像素。根据其上下文,边界 BAB 中的每个像素赋予  $2^9=512$  种可能性的一个。然后应用 CAE 算法。

$16 \times 16$  二值图最初包含 256 位信息,把它压缩成一个浮点数是非常节省的。

以上的 CAE 方式是无损的! MPEG-4 组织也考察了以上形状编码的一些简单的有损版本,如在算术编码前,二元 $\alpha$ 图可以简单地使用因子 2 或 4 进行二次采样。其代价当然是形状的质量下降。

## 2. 灰度形状编码

MPEG-4 中的术语“灰度形状编码”可能会让人误解,因为真正的形状信息编码在二元 $\alpha$ 图中。这里的灰度用来描述形状的透明度而不是纹理!

除了 RGB 帧缓冲区的位平面外,光栅图形学为 $\alpha$ 图使用特别的位平面,它用于描述图形对象的透明度。当 $\alpha$ 图拥有多个位平面时,即可引入多级透明度,例如 0 表示透明,255 表示不透明,0~255 间的任意数字表示其间的透明度。在 MPEG-4 中,对透明度编码使用灰度这个术语仅仅是因为表示透明度的数字介于 0~255 间,这与传统的 8 位灰度级相同。

MPEG-4 中的灰度形状编码使用与前述的纹理编码相同的技术。它使用 $\alpha$ 图和基于块的运动补偿,并通过 DCT 对预测误差进行编码。前面描述过,边界宏块需要进行填充,因为并非所有的像素都在 VOP 中。

透明度信息编码(灰度形状编码)是有损的,这与二元形状信息编码不同,它在默认情况下是无损的。

### 12.2.5 静态纹理编码

MPEG-4 对静态对象的纹理使用小波编码,这非常适合将纹理映射到 3D 表面的情况。

第 8 章中介绍过,小波编码可以将一幅图递归地分解为多重频率的子带。嵌入式零树小波(EZW)算法[13]通过削去子带中大量不重要的系数来提供更为紧凑的表示。

MPEG-4 静态纹理编码中的子带编码的操作过程如下所示:

- 使用 DPCM 对最低频率的子带进行编码。每个系数的预测基于其三个邻居。

- 其余子带的编码基于多级零树小波编码方式。

多级零树对最低频率子带的每一个系数拥有一个父子关系 (parent-child relation, PCR) 树, 这样可以更好地查询所有系数的位置信息。

除了初始的系数幅度, 量化的程度也对数据率有所影响。如果在量化后系数的幅度为 0, 它就被当作不重要的系数。首先, 使用一个大的量化器, 只选择最重要的系数, 然后使用算术编码方式进行编码。量化后的系数和初始的系数的差值保存在保留下的子带中, 它将在下一次使用小一点的量化器的迭代中被编码。这个过程可以不断重复, 因此具有很大的可扩展性。

346

### 12.2.6 子图像编码

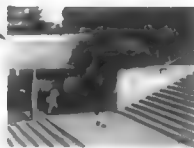
视频摄影常常涉及相机运动, 如镜头旋转、倾斜、伸缩等, 一般情况下主要是为了跟踪和检查前景 (运动的) 对象。在这种环境下, 背景可以视为静止图像。这创建了一种新的 VO 类型, 子图像——一种可以在一幅大的图像或多幅图像中自由移动的图形图像。

为了从背景中分离前景对象, 我们需要介绍子图像全景的概念——在一系列视频帧中用一幅静态图像描述静态背景, 可以使用图像缝合和卷绕技术[14]获得。大的子图像全景图像可以在视频序列的开始编码并且只送入解码器一次。当解码器接收到分别解码后的背景对象和描述相机运动的参数后, 能高效地重构场景。

图 12-10a 显示了一个子图像, 它是由一系列视频帧缝合成的全景图像。通过合并子图像背景和图 12-10b 中的风笛手, 新的视频场景 (图 12-10c) 通过子图像编码和附加的镜头旋转/倾斜和缩放参数可以很容易地进行解码。显然, 前景对象可以来自初始视频场景, 也可以创建这些前景对象以实现 MPEG-4 视频灵活的基于对象的合成。



a) 背景的子图像全景图



b) 蓝屏图中的前景对象 (风笛手) c) 合成的视频场景 (彩色插页中也有此图)。  
风笛手图像来自 Simon Fraser University  
风笛乐队, 在此感谢

图 12-10 子图像编码

347

### 12.2.7 全局运动补偿

常见的相机运动 (如镜头旋转、倾斜、伸缩 (因为它们应用于各个块, 也被称为全局运动)) 常常会造成连续视频帧间内容的快速变化。传统的基于块的运动补偿将产生大量重要的运动向量。而且, 这些相机的运动无法全部由传统的使用基于块的运动补偿所使用的运动模型描述。全



局运动补偿 (Global motion compensation, GMC) 用于解决这个问题。GMC 有 4 个主要的部分:

- **全局运动估计** 对于子图像全局运动估计计算当前图像的运动。“全局”表示由于相机运动 (镜头移动、伸缩等) 造成的总体变化。它是通过最小化子图像  $S$  和全局补偿图像  $I'$  的平方差的和得到的。

$$E = \sum_{i=1}^N (S(x_i, y_i) - I'(x'_i, y'_i))^2 \quad (12.4)$$

其基本思想是, 如果背景 (可能是缝合的) 图像是子图像  $S(x_i, y_i)$ , 我们希望新的帧主要通过全局相机运动修改的同样的背景形成。为进一步限制全局运动估计问题, 在整个图像上的运动通过如下定义的 8 个参数的预测运动模型进行参数化:

$$\begin{aligned} x'_i &= \frac{a_0 + a_1 x_i + a_2 y_i}{a_6 x_i + a_7 y_i + 1} \\ y'_i &= \frac{a_3 + a_4 x_i + a_5 y_i}{a_6 x_i + a_7 y_i + 1} \end{aligned} \quad (12.5)$$

这构成了一个约束最小化问题, 可以通过梯度下降算法[15]解决。

- **卷绕和混合** 一旦运动参数计算出来, 背景图像会与子图像缠绕到一起。卷绕后的图像坐标由式 12.5 得到。此后, 缠绕的图像混合到当前子图像, 从而生成新的子图像。这可以使用简单的平均或者某种形式的加权平均来完成。
- **运动轨迹编码** 我们不直接传递运动参数, 而只是对参考点的位移进行编码, 这被称为轨迹编码[15]。VOP 边界框角的点被选作参考点, 它们在子图像中相应的点会计算出来。这两个实体之间的差异作为差分运动向量编码并且传递。
- **选择局部运动补偿 (LMC) 或 GMC** 最后, 必须决定选择 GMC 还是 LMC。我们可以将 GMC 应用于运动的背景, 将 LMC 应用于前景。推断可知 (跳过很多细节), 如果  $SAD_{GMC} < SAD_{LMC}$ , 则选择 GMC 生成预测参考 VOP。否则像以前一样使用 LMC。

348

## 12.3 MPEG-4 的合成对象编码

计算机图形学和动画软件中所创建的视频对象不断增多, 这些对象称为合成对象, 通常和游戏、电视广告、电视节目、动画和故事片中的自然对象和场景一起展示。

在这一节中, 我们主要讨论合成对象的二维基于网格的编码和三维基于模型的编码以及动画方法。Beek、Petajan 和 Ostermann[16]对这个主题进行了更详细的论述。

### 12.3.1 2D 网格对象编码

2D 网格 (2D mesh) 是由多边形组成的二维平面区域的一小部分。这些多边形的顶点称为网格的节点。大多数常用的网格都是三角形的网格, 其中所有的多边形都是三角形。MPEG-4 标准使用两种类型的 2D 网格: 均匀网格和 Delaunay 网格[17]。这两种网格都是三角形的网格, 可以用于为自然视频对象建模, 也能用于为合成动画对象建模。

由于三角形结构 (节点之间的边) 比较简单, 可以很容易地被解码器重建, 所以在位流中不需要进行明确的编码。因此, 2D 网格的对象编码 (object coding) 比较简洁。网格中所有坐标值都以半像素精度进行编码。

每一个 2D 网格被看作一个网格对象平面 (MOP)。图 12-11 阐述了 2D MOP 的编码过程。编码可以分为几何编码 (geometry coding) 和运动编码 (motion coding)。如图所示, 输入数据是网

格中所有节点的坐标 $(x, y)$ 和三角形 $(t_m)$ 。输出数据是位移 $(dx_n, dy_n)$ 和运动的预测误差 $(ex_n, ey_n)$ ，下面将分别介绍。

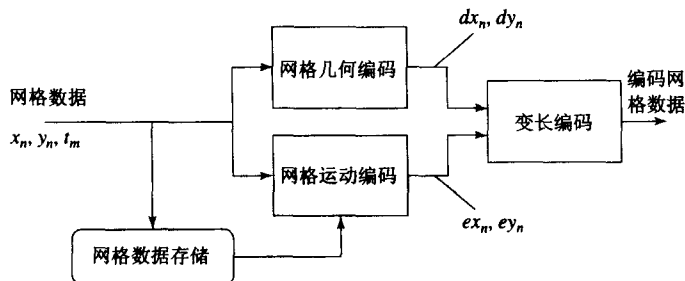


图 12-11 2D (MOP) 的编码过程

### 1. 2D 网格几何编码

349

MPEG-4 允许 4 种不同三角形结构的均匀网格。图 12-12 是具有  $4 \times 5$  网格节点的网格。每一个均匀网格可以用 5 个参数确定，前两个参数分别指定每一行和每一列的节点数；接下来的两个参数分别指定每一个矩形的水平和垂直方向上的大小；最后一个参数指定均匀网格的类型。

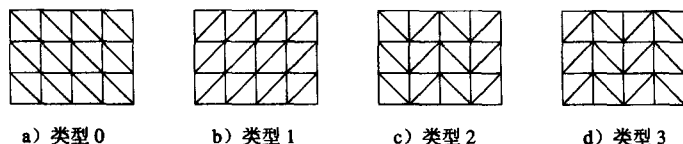


图 12-12 均匀网格的 4 种类型

均匀网格比较简单，在表现 2D 矩形对象（如整个视频帧）时效果很好。当用在任意形状的对象上时，它们被视频对象平面的边框所覆盖，影响了效率。

Delaunay 网格是能够更好地表示任意形状的二维对象的基于对象的网格。

**定义 1** 如果  $D$  是一个 Delaunay 三角网，那么它的任何一个三角形  $t_n = (P_i, P_j, P_k) \in D$  满足如下性质，即  $t_n$  的外接圆不包括其内部其他任何节点  $P_l$ 。

视频对象的 Delaunay 网格可以通过下面步骤来获得：

- 1) **选择网格的边界节点。**用多边形模拟对象的边界，多边形的顶点就是 Delaunay 网格的边界节点。一种可行的方法是选择曲率高的边界点作为边界节点。
- 2) **选择内部节点。**对象边界内的特征点，如边界点或者角点，都可以被选为网格的内部节点。
- 3) **完成 Delaunay 三角网。**一个受限的 Delaunay 三角网在边界和内部节点上起作用，多边形边界作为限制。这个三角网用线段将连续的边界节点连接起来作为边，并只在边界内部形成三角形。

### 2. 受限的 Delaunay 三角网

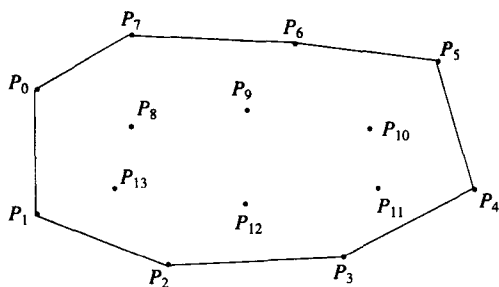
首先添加内部的边形成新的三角形。该算法会检查每一个内部边已确认它是否是本地的 Delaunay。假设两个三角形  $(P_i, P_j, P_k)$  和  $(P_j, P_k, P_l)$  共用一个边  $\overline{jk}$ ，如果  $(P_i, P_j, P_k)$  在外接圆的内部包含  $P_l$  或者  $(P_j, P_k, P_l)$  在外接圆的内部包含  $P_i$ ，那么  $\overline{jk}$  不是本地的 Delaunay，并将被边  $\overline{il}$  代替。

如果  $P_l$  正好在  $(P_i, P_j, P_k)$  的外接圆上（因此， $P_i$  也刚好在  $(P_j, P_k, P_l)$  的外接圆上）， $\overline{jk}$  将被看作局部的 Delaunay 当且仅当  $P_l$  或者  $P_i$  在四个节点中  $x$  坐标最大。

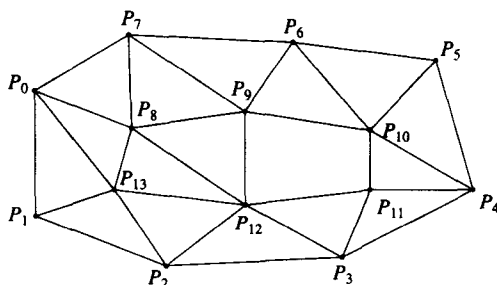
图 12-13a 和 12-13b 说明了 Delaunay 网格节点的设置和受限 Delaunay 三角形的结果。如果节点的总数是  $N$ ，并且  $N = N_b + N_i$ （其中  $N_b$  和  $N_i$  分别表示边界点和内部节点的数量），那么在

Delaunay 网格中三角形的总数为  $N_b + 2N_i - 2$ 。在上面的图中, 总数为  $8 + 2 \times 6 - 2 = 18$ 。

350



a) 边界节点 ( $P_0 \sim P_7$ ) 和内部节点 ( $P_8 \sim P_{13}$ )



b) 通过受限的 Delaunay 三角网得到三角形网格

图 12-13 Delaunay 网格

和均匀网格不同, Delaunay 网格中的节点位置是不规则的。因此, 它们必须被编码。按照 MPEG-4 的规定, 左上方的边界节点<sup>⊙</sup>的位置  $(x_0, y_0)$  必须第一个被编码, 然后按顺时针或者逆时针 (参见图 12-13a)) 方向为其他边界点编码。内部节点位置的编码可以以任何顺序进行编码。

除了第一个位置  $(x_0, y_0)$  外, 接下来的所有坐标都以差分的形式进行编码, 即对  $n \geq 1$ , 有

$$dx_n = x_n - x_{n-1}, \quad dy_n = y_n - y_{n-1} \quad (12.6)$$

之后, 对  $dx_n$  和  $dy_n$  进行变长编码。

351

### 3. 2D 网格运动编码

均匀网格和 Delaunay 网格中的每个 MOP 的运动都用它的 3 个顶节点的运动向量来描述。一个新的网格结构只能够在帧内创建, 它的三角形拓扑结构在接下来的帧中不再变化。这增强了 2D 网格运动估计的一对一映射。

对任何一个 MOP 三角形  $(P_i, P_j, P_k)$ , 如果  $P_i$  和  $P_j$  的运动向量为  $MV_i$  和  $MV_j$ , 那么为  $P_k$  的运动向量进行的预测 (值为  $Pred_k$ ) 可四舍五入为半像素精度:

$$Pred_k = 0.5 \cdot (MV_i + MV_j) \quad (12.7)$$

预测误差  $e_k$  编码为:

$$e_k = MV_k - pred_k \quad (12.8)$$

一旦对第一个 MOP 三角形  $t_0$  的三个运动向量进行编码, 至少有一个邻居 MOP 与  $t_0$  共用一条边, 它的第三个顶节点的运动向量将被编码, 以此类推。

⊙ 左上边界节点定义为有最小  $x+y$  坐标值的节点。若有多于一个的边界节点有相同的  $x+y$ , 就选  $y$  值最小的节点。



它最多有 6 个自由度, 由参数  $a_{11}$ 、 $a_{21}$ 、 $a_{31}$ 、 $a_{12}$ 、 $a_{22}$  和  $a_{32}$  来表示。

下面的  $3 \times 3$  矩阵是仿射变换的变换矩阵  $(T_x, T_y)$ , 逆时针转动  $\theta$ , 缩放倍数分别为  $S_x$ 、 $S_y$ :

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}, \quad \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

接下来的仿射变换是分别沿  $x$  轴和  $y$  轴进行倾斜:

$$\begin{bmatrix} 1 & 0 & 0 \\ H_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & H_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

其中  $H_x$  和  $H_y$  是常数, 决定倾斜的程度。

353

上面简单的仿射变换可以合成 (通过矩阵相乘) 产生复合的仿射变换。比如, 旋转后进行平移, 或者进行其他任何变换后倾斜。

可以证明 (参见习题 7), 任何复合的变换的产生将有完全相同的矩阵形式, 并且最多有 6 个自由度, 用  $a_{11}$ 、 $a_{21}$ 、 $a_{31}$ 、 $a_{12}$ 、 $a_{22}$  和  $a_{32}$  来指定。

如果目标 MOP 的三角形为

$$(P_0, P_1, P_2) = ((x_0, y_0), (x_1, y_1), (x_2, y_2))$$

参考 MOP 中对应的三角形为

$$(P'_0, P'_1, P'_2) = ((x'_0, y'_0), (x'_1, y'_1), (x'_2, y'_2)),$$

两个三角形间的映射可以定义为下式:

$$\begin{bmatrix} x'_0 & y'_0 & 1 \\ x'_1 & y'_1 & 1 \\ x'_2 & y'_2 & 1 \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix} \quad (12.13)$$

式 12.13 包含六个线性方程 (三个方程和  $x$  有关, 三个方程和  $y$  有关), 这 6 个方程用于求解未知的系数  $a_{11}$ 、 $a_{21}$ 、 $a_{31}$ 、 $a_{12}$ 、 $a_{22}$  和  $a_{32}$ 。设式 12.13 可以表示为  $X' = XA$ 。由此可知  $A = X^{-1}X'$ , 逆矩阵  $X^{-1} = \text{adj}(X) / \det(X)$ , 其中  $\text{adj}(X)$  是  $X$  的伴随矩阵,  $\det(X)$  是行列式。因此,

$$\begin{aligned} \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix} &= \begin{bmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x'_0 & y'_0 & 1 \\ x'_1 & y'_1 & 1 \\ x'_2 & y'_2 & 1 \end{bmatrix} \\ &= \frac{1}{\det(X)} \begin{bmatrix} y_1 - y_2 & y_2 - y_0 & y_0 - y_1 \\ x_2 - x_1 & x_0 - x_2 & x_1 - x_0 \\ x_1 y_2 - x_2 y_1 & x_2 y_0 - x_0 y_2 & x_0 y_1 - x_1 y_0 \end{bmatrix} \begin{bmatrix} x'_0 & y'_0 & 1 \\ x'_1 & y'_1 & 1 \\ x'_2 & y'_2 & 1 \end{bmatrix} \end{aligned} \quad (12.14)$$

其中,  $\det(X) = x_0(y_1 - y_2) - y_0(x_1 - x_2) + (x_1 y_2 - x_2 y_1)$ 。

因为网格三角形中的三个顶点都不共线, 它确保  $X$  不是奇异的, 即  $\det(X) \neq 0$ 。因此式 12.14 总是有唯一解。

上面的仿射变换都是分段的, 即每一个三角形有自己的仿射变换。当对象在动画序列中缓慢变化时, 仿射变换的效果是非常好的。图 12-15a 显示了一个简单的单词映射到 Delaunay 网格上

的情况。图 12-15b 说明在仿射变换后在动画序列中接下来的 MOP 上扭曲了的单词。



图 12-15 2D 对象动画中基于网格的纹理映射

### 12.3.2 基于模型的 3D 编码

由于人脸和身体频繁地出现于视频中, MPEG-4 为人脸对象和身体对象定义了专门的 3D 模型。使用这些新的视频对象的应用包括远程会议、人机交互、游戏以及电子商务。过去, 3D 动画研究 3D 线框模型及其动画[19], MPEG-4 比线框模型更加先进, 可以在人脸对象和身体对象的表面加阴影或进行纹理映射。

#### 1. 人脸对象编码和动画

各个人脸的人脸模型可以手工创建, 也可以通过计算机视觉或模式识别技术自动生成。但是, 前者比较麻烦而且效果不佳, 而后者在可靠性上还有所欠缺。

MPEG-4 采用由虚拟现实建模语言 (Virtual Reality Modeling Language, VRML) 组织[20]开发的通用默认人脸模型。可以指定脸部动画参数 (Face Animation Parameter, FAP) 以获取所需的动画, 它与最初的“中性”的人脸不同。此外, 还可以指定脸部定义参数 (Face Definition Parameter, FDP) 以更好地描述人脸。图 12-16 显示了 FDP 的特征点。可以被动画 (FAP) 影响的特征点用实心圆点表示, 不受动画影响的由空心圆点表示。

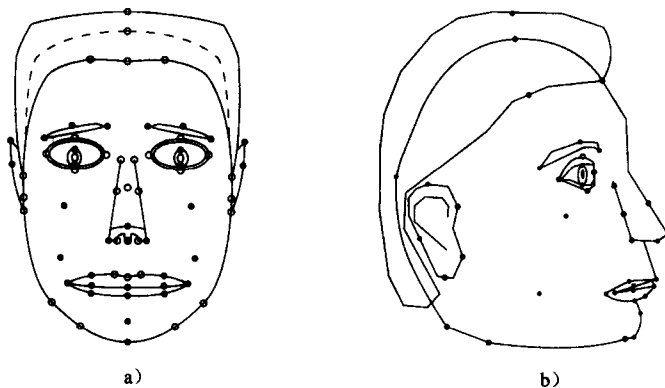


图 12-16 人脸定义参数中的特征点 (牙齿和舌头的特征点没有标出)

VRWL 组织定义了 68 种 FAP[16], FAP1 定义了嘴型, FAP2 定义了面部表情。通过对说话者当前嘴部位置建模, 嘴型 FAP 可以高度真实地为嘴唇运动编码; 其他所有的 FAP 都是为了反映头、下颏、唇、眼睑、眼球、眉毛、瞳孔、下巴、脸颊、舌头、鼻子、耳朵等部位的可能运动。

比如, 表情包含中性、喜悦、悲伤、愤怒、恐惧、厌恶和惊奇。每一种表情都能通过一组特征集来表示。以悲伤为例, 可以通过微闭的眼睛, 松弛的嘴和上扬的眉毛表示。运动的 FAP 包括

纵摇头 (head-pitch)、平摇头 (head-yaw)、转摇头 (head-roll)、张口 (open-jaw)、伸颌 (thrust-jaw)、移颌 (shift-jaw)、伸下嘴唇 (push-bottom-lip)、伸上嘴唇 (push-top-lip) 等。

为进行压缩, FAP 使用预测编码。在目标帧中对 FAP 的预测基于前一帧中的 FAP, 预测误差应用算术编码。可以使用 DCT 改进压缩率, 尽管这样做会增加计算代价。FAP 也需要量化, 通过使用不同的量化步长, 以找出某些 FAP (如张口), 可以比其他 FAP (如伸上嘴唇) 要求更低的精度。

## 2. 身体对象编码和动画

MPEG-4 版本 2 引入了身体对象, 它是对脸部对象的自然扩展。

与人类动画 (Humanoid Animation, H-Anim) 组在 VRML 联盟工作, MPEG 采用了带默认姿态的通用虚拟人体。默认的姿态是站立、双足向前、手臂在两边、手掌向内。有 296 个身体动画参数 (Body Animation Parameter, BAP)。当应用到任何与 MPEG-4 兼容的通用身体时, 它们都将产生相同的动画。

大量的 BAP 描述了身体不同部分的连接角度, 包括脊柱、肩、锁骨、肘、手腕、手指、臀、膝盖、踝及脚趾。这为身体生成了 186 个自由度, 每只手有 25 个自由度。而且, 一些身体运动可以通过多种级别来确定。例如, 根据动画的复杂度, 脊柱有 5 个不同级别, 分别支持 9、24、42、60、72 个自由度。

对特定的身体, 可以为身体维度、身体表面形状及纹理信息指定身体定义参数 (Body Definition Parameter, BDP)。身体表面形状使用一个 3D 多边形网格 (3D poly mesh) 表现, 包括 3D 空间中一个多边形平面表面集[18]。在计算机图形学中使用 3D 网格为表面建模是很流行的。与纹理映射一起使用, 它可以获得很好的 (成像逼真的) 表示效果。

BAP 的编码与 FAP 的编码相似, 即使用量化和预测编码, 预测误差通过算术编码进一步压缩。

## 12.4 MPEG-4 对象类型、规格和等级

与 MPEG-2 类似, MPEG-4 为不同应用定义了很多规格 (Profile) 和等级 (Level)。MPEG-4 中对规格和等级进行标准化主要有两个目的: 一是保证实现间的互操作性, 二是能对标准进行验证测试。MPEG-4 不仅定义了视觉规格 ([2]的第 2 部分) 和音频规格 ([2]的第 3 部分), 还定义了图形规格、场景描述规格, 而且在其系统部分 ([2]的第 1 部分) 中的一个对象描述器规格。本节中我们将对视觉规格做简要描述。

由于 MPEG-4 场景通常包含多个视频对象, 因此引入对象类型的概念, 以定义用于创建视频对象的工具以及在场景中将它们结合的方法。表 12-1 列出了应用于 MPEG-4 自然视觉对象类型<sup>①</sup>的各种工具。例如, “核心”对象类型仅用到 5 个工具。像“灰度级形状编码”、“子图像”、“隔行扫描”等工具都没有用到。

表 12-2 列出了 MPEG-4 中对对象类型的视觉规格。例如, 主要规格仅支持“简单”、“核心”、“主要”, 以及“可扩展静态纹理”这几种对象类型。

表 12-3 列出了三种最常用的视觉规格所支持的等级: 简单、核心以及主要。例如, 虽然四种不同的等级 (简单规格中的等级 2 和 3, 核心规格中的等级 2, 主要规格中的等级 1 均支持 CIF (352×288)), 但其码率以及对象的最大个数都不相同。因此, CIF 视频质量会不同。

① 我们没有列出 MPEG-4 合成视觉对象类型, 其中包括动画 2D 网格, 简单人脸、简单身体等。

表 12-1   用于 MPEG-4 自然视觉对象类型的工具

工   具	对象类型					
	简单	核心	主要	N-bit	简单扩展	扩展的静态纹理
基于 MC 的基本工具	*	*	*	*	*	
B-VOP		*	*	*	*	
二元形状编码		*	*		*	
灰度级形状编码			*			
Sprite			*			
隔行扫描			*			
时间可扩展性 (P-VOP)		*	*		*	
空间和时间可扩展性 (矩形 VOP)				*		
N-bit					*	
扩展的静态纹理						*
差错修复	*	*	*	*	*	

表 12-2   MPEG-4 自然视觉对象类型及规格

对象类型	规   范					
	简单	核心	主要	简单扩展	N-bit	扩展的静态
简单	*	*	*	*	*	
核心		*	*		*	
主要			*			
简单扩展				*		
N-bit					*	
扩展的静态纹理			*			*

表 12-3   MPEG-4 简单、核心以及主要视觉规格中的等级

规   范	级	典型的图片尺寸	码率 (bps)	最大对象数
简   单	1	176×144 (QCIF)	64k	4
	2	352×288 (CIF)	128 k	4
	3	352×288 (CIF)	384 k	4
核   心	1	176×144 (QCIF)	384k	4
	2	352×288 (CIF)	2M	16
主   要	1	352×288 (CIF)	2M	16
	2	720×576 (CCIR601)	15M	32
	3	1920×1080 (HDTV)	38.4M	32

12.5   MPEG-4 Part10/H.264

ISO/IEC MPEG 的联合视频小组 (Joint Video Team, JVT) 和 ITU-T 视频编码专家组 (Video Coding Experts Group, VCEG) 开发了 H.264 视频压缩标准, 该标准在 2003 年 3 月完成。其最初的题目是 “H.26L”。使用基于这个新标准的软件的前期研究表明, H.264 可以提供比 MPEG-2 高 50% 的压缩率, 提供比 H.263+ 和 MPEG-4 的高级简单规格高 30% 的压缩率。

这项工作的成果是产生两个相同的标准: ISO MPEG-4 Part10 以及 ITU-T H.264。H.264 在压缩性能上优于 MPEG-2 很多, 在许多应用中成为传输 HDTV 视频内容的主要候选方案之一。下一



节将依据[21]来介绍这个新标准。

### 12.5.1 核心特征

与之前 ITU-T H.263+类似, H.264 详述了一个基于块的运动补偿转换混合解码器, 其中有 5 个主要块:

- 熵解码。
- 运动补偿或帧内预测。
- 反向扫描、量化和剩余像素的转换。
- 重构。
- 重构像素上的循环内去块效应滤波器。

每张图片可以再分为宏块 (16×16 的块), 而且任意大小的宏块片可以聚合成多个宏块为自包含的单元。

#### 1. 基于 VLC 的熵解码

在变长熵解码器中用到了两种熵方法: 统一变长编码 (UVLC) 和上下文自适应变长编码 (CAVLC)。UVLC 使用简单指数 Golomb 编码来对头部数据、运动向量和其他非剩余数据进行解码, 而对于剩余系数则用较复杂的 CAVLC 解码。

在 CAVLC 中, 为每种数据类型 (runs、levels 等) 预定义了多个 VLC 表, 预定义规则基于上下文 (之前已解码的符号) 预测最佳 VLC 表。CAVLC 允许对每种数据类型使用多个统计模型, 而且比起现有的固定 VLC (如 H.263+), CAVLC 改善了熵编码的效率。

#### 2. 运动补偿 (P 预测)

H.264 中的帧间运动补偿与 H.263+类似, 但更为复杂。和 H.263+中限制运动补偿块的大小只能为 16×16 或者 8×8 不同, H.264 使用一种树型结构运动分割将块分割至 4×4 大小 (16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4)。这使运动对象的运动补偿更精确。

而且, H.264 中的运动向量可以达到采样精度。一个 6 级接收滤波器用来进行半像素过滤, 以保留高频。对 1/4 像素过滤取平均, 不仅可以提供比半像素更精确的运动, 而且可提供更低通的滤波。多参考帧也是 H.264 的一个标准特征, 所以在它的所有规格中, 可为每个宏块选择一个不同的参考帧。

#### 3. 帧内预测 (I 预测)

比起之前的视频标准 (如 H.263+), H.264 利用了更多的空间预测。帧内编码的宏块都是使用邻近的重构像素 (已进行帧间和帧内编码的重构像素) 预测得到。与运动补偿类似, 可以为每个已进行帧内编码的宏块选择不同的块大小 (16×16 或 4×4)。对于 4×4 的块有九种预测模式 (宏块中每个 4×4 的块可以有不同的预测模式), 16×16 的块有 4 种预测模式。这种较复杂的帧内预测能力很强, 它可以在时间预测失败时极大地减少传输的数据量。

#### 4. 转换、扫描和量化

假设在 H.264 中有功能强大和精确的 P 预测和 I 预测方案, 那么剩余像素的空间相关性将非常小。因此, 一个简单整数精度 4×4 DCT 可以有效地节省能量。整数算法在所有处理器上可以进行精确的反变换, 并且减少前述基于变换的编/解码中编码器/解码器不匹配的问题。H.264 还提供了一个具有非线性步长的量化方案, 可以在量化尺度的高端和低端获得精确的速率控制。

#### 5. 循环内去块效应滤波

H.264 详述了一种复杂的信号自适应去块效应滤波器, 其中一组滤波器应用到 4×4 块的边界。滤波器长度、强度和类型 (去块/平滑) 根据宏块编码参数 (帧内或者帧间编码、运动向量差、参考帧差、系数编码) 和空间活动 (边界检测) 的不同而不同, 这就使方块现象在不扭曲视觉特征

的情况下被消除了。H.264 去块效应滤波对于增强标准的主观质量很有效。

### 12.5.2 基本规格特征

H.264 的基本规格 (Baseline profile) 主要用于实时对话应用, 如视频会议。它包含前面所讨论的 H.264 中所有的核心编码工具以及下面介绍的容错工具, 适用于易出错的传输 (如 IP 和无线网络):

- **任意宏块片顺序 (Arbitrary slice order, ASO)** 一张图片内宏块片的解码顺序不一定遵循单调递增的顺序。它可以对包交换网络中的无序包进行解码, 这样可以减少延迟。
- **灵活的宏块顺序 (Flexible macroblock order, FMO)** 宏块可以以任何顺序进行解码, 像棋盘式的交替模式而不是光栅扫描模式。这对于易出错的网络很适用, 这样由于宏块片的丢失所导致的图中宏块的丢失就比较分散, 不容易被人眼察觉。这个特征对于减少抖动和延时很有效, 因为解码器可以决定不等待迟到的宏块片, 并且同样可以生成可接受的图片。
- **冗余的宏块片** 宏块片的冗余副本也被解码, 从而进一步改善容错性。

### 12.5.3 主要规格特征

H.264 中定义的主要规格 (Main profile) 适用于对延时要求不高的应用, 如广播和存储媒介。主要规范包含基本规格的所有特征 (ASO、FMO 及冗余的宏块片) 以及下面介绍的对延时要求低和复杂度较高的特征, 从而获得最高压缩率:

- **B 宏块片** H.264 中的双向预测模式比现有标准中的预测模式要灵活。经过双向预测的图片可以被用做参考帧。每个宏块的两个参考帧可以是在任意的时间方向的, 只要它们在当前的参考帧缓冲区中可用。因此, 除了通常的前向+后向双向预测外, 还可以有后向+后向或者前向+前向预测。
- **上下文自适应二进制算术编码 (Context Adaptive Binary Arithmetic Coding, CABAC)** 这种编码模式用二进制算术编码代替了基于 VLC 的熵编码, 它针对不同的数据类型和上下文使用不同的自适应统计模型。
- **加权预测** 对于每个宏块片都指定了用来修改运动补偿预测样本的全局加权值 (乘数和一个偏移), 用来预测微小的改变以及其他全局性的影响, 如淡入淡出。

360

### 12.5.4 扩展规格特征

扩展规格 (eXtended profile, 或 X 规范) 主要用于新的视频流应用。这个规范具有延时要求低的特征、位流交换的特征, 以及更多的容错工具。它包含基本规格所有特征及下列特征:

- **B 宏块片。**
- **加权预测。**
- **宏块片数据分割** 它将具有不同重要性的宏块片数据分割为不同的序列 (头部信息、剩余信息), 以便重要的数据可以在更可靠的信道上传输。
- **SP 和 SI 宏块片类型** 这些宏块片包含特殊的时间预测模式、允许位流交换、快进/快退以及随机访问。

H.264 核心特征的极大改进, 以及新的编码工具使它比现有的 ITU-T 和 MPEG 标准在压缩率、容错和主观质量上有很大的改善。

## 12.6 MPEG-7

随着越来越多的多媒体内容成为众多应用的不可或缺部分, 有效性以及高效的信息检索成为首要考虑的问题。1996 年 10 月, MPEG 继 MPEG-1、2 和 4 之后开始进行另一个主要标准 MPEG-7 的开发。

MPEG-4 与 MPEG-7 的一个相同之处在于它们都关注的是视听对象。MPEG-7 的一个主要目

标[22]是满足像数字图书馆这类应用中对于基于视听内容的检索（或视听对象检索）的需要。然而，对于信息检索没有做限制——它可以应用于任何多媒体应用，包括多媒体数据的产生（内容创建）和使用（内容消耗）。

MPEG-7 在 2001 年 9 月成为一个国际标准。在 ISO/IEC 15938 文档[23]中，它的正式名称为多媒体内容描述接口（Multimedia Content Description Interface）。该标准由七部分组成，即系统、描述定义语言、视频、音频、多媒体描述方案、参考软件以及一致性。

MPEG-7 支持多种多媒体应用。它的数据可以包括静态图像、图形、3D 模型、音频、语音、视频以及复合信息（这些元素如何组合）。这些 MPEG-7 数据元素可以用文本或二进制格式（或者两者兼用来表示）。第一部分（系统）定义了 MPEG-7 二进制格式（BiM）数据的语法。第二部分（描述定义语言）描述了采用 XML Schema 作为其语言的文本格式的语法。在文本格式与二进制格式表示之间定义了双向无损映射。

361

图 12-17 给出了一些可得益于 MPEG-7 的应用。如图所示，特征可取自 MPEG-7 描述。接着它们被 MPEG-7 编码器编码并发送到存储和传输媒体。多种搜索和查询引擎处理搜索和浏览请求，这由 Internet 的一系列拉式（pull）动作组成，同时，过滤代理（filter agent）过滤出许多材料并推（push）到终端——用户或消费数据的计算机系统及应用。

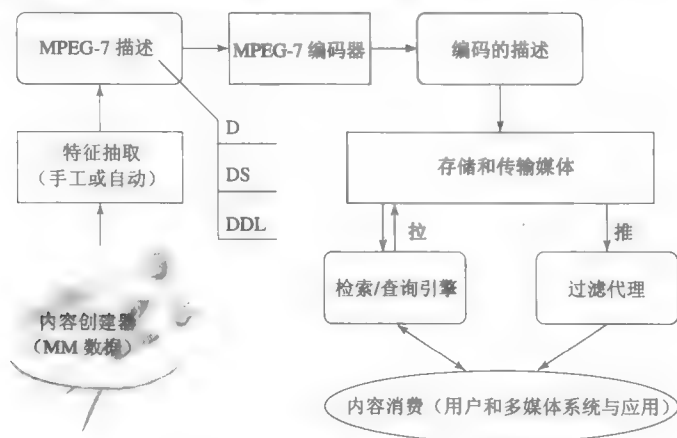


图 12-17 使用 MPEG-7 的可能应用

为了进行多媒体内容描述，MPEG-7 还开发了描述子（Descriptor, D）、描述方案（Description Scheme, DS）和描述定义语言（Description Definition Language, DDL）。下面是一些重要概念：

- **特征** 指数据的特性。
- **描述子（D）** 特征的定义（语法和语义）。
- **描述方案（DS）** 描述子和描述模式间结构和关系的规范（参见[24]）。
- **描述** 一组实例化的描述子的和描述模式，描述了内容、存储以及内容使用等的结构和概念信息。
- **描述定义语言（DDL）** 表达和组合描述模式以及描述子的语法规则（参见[25]）。

可见[23]，MPEG-7 的范畴就是对用于描述的描述子、描述模式和描述定义语言进行标准化，而产生和使用描述的机制和处理过程超出了 MPEG-7 的范畴。这个问题还有待解决，它有利于工业界改革创新和公平竞争，更重要的是可以推动新技术的不断进步。

362

与 MPEG-1 视频中的仿真模型（SM）、MPEG-2 视频中的测试模型（TM），以及 MPEG-4（视频、

音频、SNHC 和系统)中的验证模型 (VM) 类似, MPEG-7 把它的工作模型命名为实验模型 (Experimentation Model, XM)——一个字母顺序双关语! XM 提供了用于评价描述子、描述模式和描述定义语言的多种工具的描述, 以便于进行实验和验证, 并可在全世界对多个独立的部分进行比较。第一组这类实验称作核心实验 (core experiments)。

### 12.6.1 描述子

MPEG-7 的描述子用于描述如颜色、纹理、形状、动作这些低级特征以及如事件、抽象概念这些语义对象的高级特征。像前面所说的那样, 用于自动、甚至半自动的特征抽取方法和处理过程都不是标准中的一部分。尽管在图像、音频处理、计算机视觉以及模式识别领域做了大量工作并取得一些进展, 但自动、可靠的 (特别是高级别的) 特征提取在近期还不太可能实现。

应该根据描述子的性能、效率以及大小的比较来选择描述子, 表示基本视觉特征的低级视觉描述子[26]包括:

#### • 颜色

- **颜色空间** RGB、YcbCr、HSV (色调、饱和度、对比度)[18]、HMMD。(HueMaxMinDiff)[27]从 RGB 的一个  $3 \times 3$  矩阵推导出的 3D 颜色空间以及单色。
- **颜色量化** 线性、非线性和查找表。
- **主颜色** 每个区域或图像中少数有代表的颜色, 在基于颜色相似性的图像检索中很有用处。
- **可扩展的颜色** 在 HSV 颜色空间中的一个颜色柱状图。用 Haar 变换进行编码, 因此是可扩展的。
- **颜色分布** 为进行基于颜色分布的信息检索所做的颜色在空间上的分布。
- **颜色结构** 一个颜色结构元素的频率既描述颜色内容也描述它在图像中的结构。颜色结构元素由具有相同颜色的局部邻域中的几个图像样本组成。
- **帧组/图像组 (GoF/GoP) 颜色** 与可扩展颜色类似, 只是它应用于一个视频段或一个静态图像组。对于 GoF/GoP 中所有颜色柱状图的各个柱进行取平均、取中值或取交集操作可以得到总的颜色柱状图, 然后送至 Haar 变换。

#### • 纹理

- **同质纹理** 利用可以定量表示同质纹理区域的方向和尺度调整的 Gabor 滤波器[28]。Gabor 滤波器的优点是可以在空间和空间频率域内提供同步最佳分辨率[29]。而且, 它们是符合人类视觉特征的带通滤波器。一个滤波器堆由 30 个 Gabor 滤波器组成, 有 5 种不同的尺度, 每个尺度有 6 种不同的方向, 用来提取纹理描述子。
- **纹理浏览** 用于表示和浏览同质纹理的边的规律性、粗糙度和方向性的描述[30]。此时也要使用 Gabor 过滤器。
- **边柱状图** 表示 4 个方向 ( $0^\circ, 45^\circ, 90^\circ, 135^\circ$ ) 的边和一个无向边的空间分布。图像可以分为小的子图, 并为每个子图生成一个有 5 个柱的边界柱状图。

#### • 形状

- **基于区域的形状** 用一组 ART (Angular Radial Transform) 系数来描述一个对象的形状。一个对象可以由一个或多个区域组成, 对象中可能会有一些洞。ART 转换是根据单位圆上的极坐标而定义的二维复杂变换。ART 基本函数在角度维和径向维上是分开的。36 个基本函数 (12 个角度和 3 个径向) 用来抽取形状描述子。
- **基于轮廓的形状** 利用曲率尺度空间 (Curvature Scale Space, CCS) 表示[32]有固定的尺度和旋转, 对于非刚性运动和部分遮挡足够健壮的形状。

- **三维形状** 对三维网络模型和形状索引的描述[33]。对于整个网格的形状索引的柱状图用来作为描述子。
- **运动**
  - **摄像机运动** 固定、旋转、倾斜、转动、前后移动、跟踪、上下移动（参见图 12-18 和 [34]）。

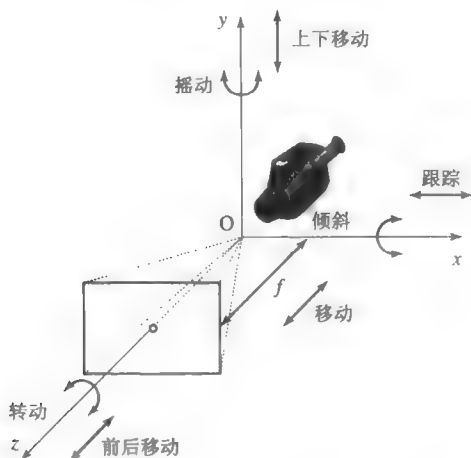


图 12-18 摄像机运动：旋转、倾斜、转动、前后移动、跟踪以及上下移动  
（摄像机有一个有效焦距  $f$ 。图中表明摄像机开始位于圆点，面朝 Z 轴方向）

- **对象运动轨道** 关键点  $(x, y, z, t)$  的列表。可选的插值函数可用来详细描述路径上的加速情况（参见[34]）。
- **变量对象运动** 基本模型是平移、旋转、缩放、偏移以及这几种组合的二维仿射模型。一个平面透视模型和二次模型可适用于透视失真以及更复杂的运动。
- **运动动作** 提供对视频的强度、速度、状态等的描述。例如，“曲棍球游戏中评分”或者“采访一个人”这样的视频。
- **定位**
  - **区域定位器** 指定一个有方形或多边形的图像中区域的定位。
  - **时空定位器** 描述视频序列中的时间空间区域。利用一组或多组区域描述子以及它们的运动。
- **其他**
  - **面部识别** 一个标准的面部图像可表示为一个一维向量，然后投影成一个有 49 个基本向量的集合，代表所有可能的面部向量。

## 12.6.2 描述方案

本节给出基本元素、内容管理、内容描述、导航和访问、内容组织以及用户交互领域中 MPEG-7 描述方案的概述。

- **基本元素**
  - **数据类型和数学结构** 向量、矩阵、柱状图等。
  - **结构** 链接媒体文件及定位段、区域等。
  - **模式工具** 包括根元素（MPEG-7XML 文档和描述的开始元素）、顶级元素（为面向特

定内容的描述组织 DS) 和打包工具 (将一个描述的相关的 DS 组件组成包)。

- 内容管理

- **媒体描述** 仅包含一个 DS, 即媒体信息 DS, 它由一个媒体标识描述子和一个或多个媒体规范描述子组成, 媒体规范描述子中包含诸如编码方法、代码转换提示、存储和传输格式等信息。
- **创建和生成描述** 包含创建 (标题、创建者、创建地点、日期等)、分类 (类型、语言、所属指引等) 以及相关材料的信息。
- **内容使用描述** 提供关于使用权限、使用记录、可用性以及经费 (生产成本、内容使用的收入) 的信息的各种 DS。

- 内容描述

- **结构化描述** 段描述方案描述内容的结构化方面。段是一个音频视频对象的一部分。段之间的关系通常用段树表示。当关系不是纯层次结构时, 可使用段图表示。

段描述方案可以用一个类对象实现。它有 5 个子类, 即视听段描述方案、音频段描述方案、静态区域描述方案、移动区域描述方案以及视频段描述方案。子类描述方案可以递归的包含其子类。

例如, 一个静态区域描述方案可以用于从图像的创建 (标题、创建者、日期)、使用 (商标)、媒体 (文件格式)、文本标注、颜色柱状图以及可能的纹理描述子等方面描述图像。这个初始区域 (在这种情况下是图像) 可以再分解为许多区域, 这些区域也可以有它们自己的描述方案。

图 12-19 描述了一个完成船只营救任务的视频段, 图中一个人从直升飞机上降落到船中。视频段中有三个运动区域。可以构建一个段图包含像视频帧 (直升飞机、人、船只) 的空间关系组合以及区域运动 (上方、上面、紧邻、运动朝向等) 的结构化描述。

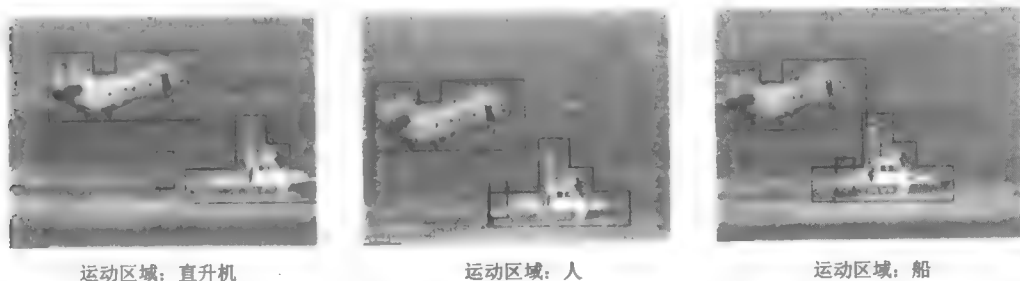


图 12-19 MPEG-7 视频段 (彩色插页中也有此图)

- **概念化描述** 包括内容的更高级别 (非结构化) 描述, 例如, 对篮球赛或湖人队球赛的事件描述模式 (Event DS)、对 John 或人的对象描述方案、特定时间或地点的语义性质的状态描述方案以及对像 “自由” 或 “秘密” 等抽象概念的概念描述方案。与段描述方案类似, 概念描述方案也可以被组织为一棵树或图。

- 导航与访问

- **摘要** 为内容的快速浏览和导航提供视频摘要, 通常只显示关键帧。支持下列描述方案: 摘要描述方案、层次摘要描述方案、重要级描述方案、连续摘要描述方案。层次摘要提供了一个含有多级的关键帧层次, 而连续摘要则提供幻灯片显示或者音频视频快进, 并且可能有音频和文本的同步。图 12-20 描述了一个公园中 “龙船” 赛视频的摘要。摘要

组织为一级层次结构。每级的每个视频段用一个指甲大小的关键帧描述。

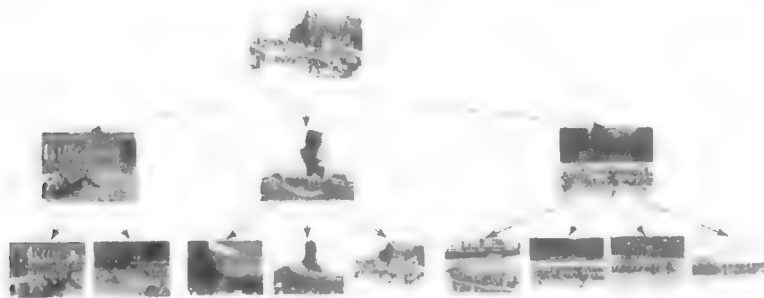


图 12-20 视频摘要

- **分割和分解** 这里是指视图的分割和分解。视图分割（由视图 DS 指定）描述视听数据的不同空间视图和频率视图，如一个空间视图（可以是一个图像的空间段），时间视图（如在一个视频的时间段中），频率视图（如在一个图像的小波子带中）或者分辨率视图（如在一个指甲大小的图中）等。视频分解 DS 指定为了组织视听数据的视图而进行的不同树或图的分解，如一个空间树 DS（一个四叉树图像分解）。
- **内容的变化** 变化 DS 描述图像分辨率、帧率、颜色削减、压缩等的原数据的变化。服务器利用它来实现以特定的 QoS 在网络和终端自适应传输视听数据。
- **内容组织**
  - **集合** 集合结构 DS 将音频视频内容组成簇。它指定了簇元素的一般性质和簇之间的关系。
  - **模型** 模型 DS 包含可能性模型 DS、分析模型 DS 以及用来抽取采集属性和特征的模型和统计的分类器 DS。
- **用户交互**
  - **用户偏好** DS 描述用户在视听内容的消耗方面的偏好，如内容类型、浏览模式、个人特征以及用户喜好是否可以被分析用户行为的代理改变。

366  
367

### 12.6.3 描述定义语言

MPEG-7 采用最初由 W3C 开发的 XML 模式语言作为描述定义语言。由于 XML 模式语言并不是专门为视听内容设计的，所以要对它进行扩展。MPEG-7DDL 大致有如下一些部分：

- **XML 模式结构部分**
  - 模式：定义和声明的封皮。
  - 主要结构部分，如简单和复杂的类型定义，以及属性和元素的声明。
  - 次要结构部分，如属性组定义、身份限制定义、组定义，以及符号声明。
  - “助手”部分，如注释、词缀，以及通配符。
- **XML 模式数据类型部分**
  - 原始和派生的数据类型。
  - 用户派生新数据类型的机制。
  - 比 XML 1.0 更好的类型检查。
- **MPEG-7 扩展**
  - 数组和矩阵数据类型。
  - 多种媒体类型，包括音频、视频以及音频视频表示。
  - 枚举数据类型 MineType、CountryCode、RegionCode、CurrencyCode 以及

368

CharacterSetCode。

- 对描述子和描述模式的知识产权管理及保护 (IPMP)。

## 12.7 MPEG-21

跨入新的世纪后, 多媒体几乎遍布各个领域。目前, 内容生产者和内容消费者的数目每日递增。然而, 直到现在也没有一个统一的标准来定义、标识、描述、管理以及保护多媒体数据。

最新标准 MPEG-21: 多媒体框架[35]从 2000 年 7 月开始开发。它的技术报告草案中称:

“MPEG-21 的目标是定义一个多媒体框架, 使大范围的网络和设备中的多媒体资源可以被不同人群透明而大规模地使用。”

MPEG-21 中有下列 7 个关键元素:

- **数字条目声明** 为声明数字条目建立的一种统一灵活的提取和互操作的模式。
- **数字条目的标识和描述** 为数字条目的标识和描述的标准化建立一个框架, 而不管它们的来源、类型或粒度。
- **内容管理和使用** 提供一个接口和协议, 使内容的管理和使用 (查询、高速缓存、存档和分配等) 更方便。
- **知识产权管理和保护 (IPMP)** 使内容能被可靠的管理和保护。
- **终端和网络** 对在大范围的网络和终端上有 QoS 要求的内容进行互操作和透明访问。
- **内容表示** 以一种适当的方式来表示内容, 以实现 MPEG-21 的目标, 即“随时随地的内容”。
- **事件报告** 为报告事件 (用户交互) 建立度量和接口, 以便理解性能和选择。

2003 年, MPEG-21 的 9 个部分中的大多数成为国际标准。MPEG-21 的开发涉及与许多国际组织以及标准化团体, 包括 W3C、MSF、SMPTE 及 DAVIC 的协作。该标准的目标是非常远大的。与 MPEG 早期的成功标准相比, 它所带来的效率及影响将指日可待。

369

## 12.8 进一步探索

Puri 和 Chen 的书[3]以及 Pereira 和 Ebrahimi 的书[36]中对 MPEG-4 进行了比较详细的介绍。Manjunath、Salembier 和 Sikora 的书[37]对 MPEG-7 进行了论述。

本章网页的 Further Exploration 部分提供了下面链接:

- MPEG 主页。
- MPEG 的 FAQ 页。
- MPEG-4 的综述、教程和工作文档。
- MPEG-4 Part10/H.264 的教程。
- MPEG-7 的概述和 MPEG-21 的工作文档。
- 构成 MPEG-7 DDL 基础的 XML 模式文档。

## 12.9 练习

1. MPEG-4 运动补偿被认为是基于 VOP (视频对象平面) 的。而最终, 为了运动补偿, VOP 仍然被分成宏块 (内部宏块、边界宏块等)。
  - (a) 当前实现的潜在问题是什么? 如何改进?
  - (b) 有真正的基于 VOP 的运动补偿吗? 与当前的实现相比如何?
2. MPEG-1、2、4 是著名的解码器标准。而压缩算法 (编码器的细节) 仍然要留待将来做改进和



开发的。对于 MPEG-4 来说, 视频对象分割的主要问题, (即如何获得 VOP) 没有具体说明。

(a) 提出一种你自己处理视频对象分割的方法。

(b) 你的方法中有哪些潜在的问题?

3. 为什么要在 MPEG-4 基于 VOP 的编码中引入填充? 说出填充存在的问题。

4. 运动向量可以是半像素精度。特别的, MPEG-4 在亮度 VOP 中允许 1/4 像素精度。描述一个可以实现这个精度的算法。

5. 编程实现计算下面  $8 \times 8$  的块的 SA-DCT:

0	0	0	0	16	0	0	0
4	0	8	16	32	16	8	0
4	0	16	32	64	32	16	0
0	0	32	64	128	64	32	0
4	0	0	32	64	32	0	0
0	16	0	0	32	0	0	0
0	0	0	0	16	0	0	0
0	0	0	0	0	0	0	0

6. 与普通 DCT 相比, SA-DCT 的计算开销有多大? 假设视频对象是一个  $8 \times 8$  块中间的一个  $4 \times 4$  的正方形。

7. 可将仿射转换组合进行复合仿射转换。证明复合仿射转换具有形式完全相同的矩阵 (最后一列为  $[0 \ 0 \ 1]^T$ ) 且最多有 6 级自由度, 用参数  $a_{11}$ 、 $a_{21}$ 、 $a_{31}$ 、 $a_{12}$ 、 $a_{22}$ 、 $a_{32}$  表示。

8. 基于网格的运动编码应用于二维动画和面部动画时的效果很好。当它应用在人体动画中时主要有哪些问题?

9. MPEG-4 是如何进行基于 VOP 的运动补偿的? 简述必要的步骤并画出块图来说明数据流向。

10. 开发 MPEG-7 的主要动机是什么? 给出现实世界的应用中得益于 MPEG-7 的三个例子。

11. “基于区域”和“基于轮廓”是 MPEG-7 中两个主要的形状描述子。显然, 描述区域和轮廓的形状有很多种方法。

(a) 你更喜欢哪种形状描述子?

(b) 与 MPEG-7 中的 ART 和 CSS 相比如何?

## 12.10 参考文献

- [1] T. Sikora, "The MPEG-4 Video Standard Verification Model," *IEEE Transactions on Circuits and Systems for Video Technology*, Special issue on MPEG-4, 7(1): 19–31, 1997.
- [2] *Information technology — Generic Coding of Audio-Visual Objects*, International Standard: ISO/IEC 14496, Parts 1–6, 1998.
- [3] A. Puri and T. Chen, eds., *Multimedia Systems, Standards, and Networks*, New York: Marcel Dekker, 2000.
- [4] G. Fernando, et al., "Java in MPEG-4 (MPEG-J)," in *Multimedia, Systems, Standards, and Networks*, A. Puri and T. Chen, eds., New York: Marcel Dekker, 2000, p. 449–460.
- [5] *Video Coding for Low Bit Rate Communication*, ITU-T Recommendation H.263, Version 1, Nov. 1995, Version 2, Feb. 1998.
- [6] A. Puri, et al., "MPEG-4 Natural Video Coding — Part I," in *Multimedia, Systems, Standards, and Networks*, A. Puri and T. Chen, eds., New York: Marcel Dekker, 2000, p. 205–244.

- [7] T. Ebrahimi, F. Dufaux, and Y. Nakaya, "MPEG-4 Natural Video Coding — Part II," in *Multimedia, Systems, Standards, and Networks*, A. Puri and T. Chen, eds., New York: Marcel Dekker, 2000, p. 245–269.
- [8] P. Kauff, et al., "Functional Coding of Video Using a Shape-Adaptive DCT Algorithm and an Object-Based Motion Prediction Toolbox," *IEEE Transactions on Circuits and Systems for Video Technology*, Special issue on MPEG-4, 7(1): 181–196, 1997.
- [9] J. Ostermann, E.S. Jang, J. Shin, and T. Chen, "Coding of Arbitrarily Shaped Video Objects in MPEG-4," in *Proceedings of the International Conference on Image Processing (ICIP '97)*, 1997.
- [10] *Standardization of Group 3 Facsimile Apparatus for Document Transmission*, ITU-T Recommendation T.4, 1980.
- [11] *Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus*, ITU-T Recommendation T.6, 1984.
- [12] *Information Technology — Coded Representation of Picture and Audio Information — Progressive Bi-Level Image Compression*, International Standard: ISO/IEC 11544, also ITU-T Recommendation T.82, 1992.
- [13] J.M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *IEEE Transactions on Signal Processing*, 41(12): 3445–3462, 1993.
- [14] G. Wolberg, *Digital Image Warping*, Los Alamitos, CA: Computer Society Press, 1990.
- [15] M.C. Lee, et al., "A Layered Video Object Coding System Using Sprite and Affine Motion Model," *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1): 130–145, 1997.
- [16] P. van Beek, "MPEG-4 Synthetic Video," in *Multimedia, Systems, Standards, and Networks*, A. Puri and T. Chen, eds., New York: Marcel Dekker, 2000, pp. 299–330.
- [17] A.M. Tekalp, P. van Beek, C. Toklu, and B. Gunsel, "2D Mesh-Based Visual Object Representation for Interactive Synthetic/Natural Digital Video," *Proceedings of the IEEE*, 86: 1029–1051, 1998.
- [18] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics: Principles and Practice*, 2nd ed., Reading, MA: Addison-Wesley, 1990.
- [19] A. Watt and M. Watt, *Advanced Animation and Rendering Techniques*. Reading MA: Addison-Wesley, 1999.
- [20] *Information Technology — The Virtual Reality Modeling Language — Part 1: Functional Specification and UTF-8 Encoding*, International Standard: ISO/IEC 14772-1, 1997.
- [21] T. Wiegand, "JVT-F100: Study of Final Committee Draft of Joint Video Specification (ITU-T Rec. H.264 — ISO/IEC 14496-10 AVC), Draft 1d," in *Sixth Meeting of JVT of ISO/IEC MPEG and ITU-T VCEG*, 2002.
- [22] S.F. Chang, T. Sikora, and A. Puri, "Overview of the MPEG-7 Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, Special issue on MPEG-7, 11(6): 688–695, 2001.
- [23] *Information Technology — Multimedia Content Description Interface*, International Standard: ISO/IEC 15938, Parts 1–6, 2001.
- [24] P. Salembier and J. R. Smith, "MPEG-7 Multimedia Description Schemes," *IEEE Transactions on Circuits Systems for Video Technology*, 11(6): 748–759, 2001.
- [25] J. Hunter and F. Nack, "An Overview of the MPEG-7 Description Definition Language (DDL) Proposals," *Signal Processing: Image Communication*, 16(1-2): 271–293, 2001.
- [26] T. Sikora, "The MPEG-7 Visual Standard for Content Description — An Overview," *IEEE Transactions on Circuits and Systems for Video Technology*, Special issue on MPEG-7, 11(6): 696–702, 2001.

- [27] B.S. Manjunath, J.-R. Ohm, V.V. Vasudevan, and A. Yamada, "Color and Texture Descriptors," *IEEE Transactions on Circuits Systems for Video Technology*, 11: 703–715, 2001.
- [28] B.S. Manjunath, G.M. Haley, and W.Y. Ma, "Multiband Techniques for Texture Classification and Segmentation," in *Handbook of Image and Video Processing*, A. Bovik, ed., San Diego: Academic Press, 2000, pp. 367–381.
- [29] T. P. Weldon, W. E. Higgins, and D. F. Dunn, "Efficient Gabor Filter Design for Texture Segmentation," *Pattern Recognition*, 29(12): 2005–2016, 1996.
- [30] P. Wu, B.S. Manjunath, S. Newsam, and H.D. Shin, "A Texture Descriptor for Browsing and Similarity Retrieval," *Signal Processing: Image Communication*, 16(1-2): 33–43, 2000.
- [31] P. Salembier and J. Smith, "Overview of MPEG-7 Multimedia Description Schemes and Schema Tools," in *Introduction to MPEG-7: Multimedia Content Description Interface*, B.S. Manjunath, P. Salembier, and T. Sikora, eds., New York: Wiley, 2002, Chapter 6.
- [32] F. Mokhtarian and A.K. Mackworth, "A Theory of Multiscale, Curvature-Based Shape Representation for Planar Curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8): 789–805, 1992.
- [33] J.J. Koenderink and A.J. van Doorn, "Surface Shape and Curvature Scales," *Image and Vision Computing*, 10: 557–565, 1992.
- [34] S. Jeannin, et al., "Motion Descriptor for Content-Based Video Representation," *Signal Processing: Image Communication*, 16(1-2): 59–85, 2000.
- [35] *Information Technology – Multimedia Framework*, International Standard: ISO/IEC 21000, Parts 1–9, 2003.
- [36] F. Pereira and T. Ebrahimi, *The MPEG-4 Book*, Upper Saddle River, NJ: Prentice Hall, 2002.
- [37] B.S. Manjunath, P. Salembier, and T. Sikora, eds., *Introduction to MPEG-7: Multimedia Content Description Interface*, New York: Wiley, 2002.

## 第 13 章 音频压缩技术基础

在多媒体系统中, 音频信息的压缩有一些特别之处。某些技术是我们所熟悉的, 而另外一些则是比较新的技术。在本章中, 我们将简单介绍语音压缩的基本技术, 这是一个具有很长历史并且非常宽泛的研究领域。更多的信息可以参考本章的 13.4 节和参考文献。

在下一章中, 我们将介绍在运动图像专家组 (MPEG) 的支持下开发的一组音频压缩工具。由于多媒体领域的读者往往会对这一部分很感兴趣, 因此我们将对该主题进行详细介绍。

首先我们将回顾第 6 章中关于多媒体数字音频的介绍, 如用于扩展数字音频的 $\mu$ 率。这经常要和利用时间冗余的简单技术结合使用。在第 10 章中我们了解到, 在视频压缩中, 当前时间和过去时间之间的信号的差值可用以有效减少信号值的大小, 更重要的是, 将像素值 (现在是差值) 的直方图聚集在一个相对较小的范围内。降低值间差异的结果是大大减少熵值, 这样后续的赫夫曼编码就可以产生具有较高压缩率的位流。

同样的道理也可以应用在这里。回顾一下, 第 6 章中介绍过, 量化后的采样输出称为脉冲编码调制 (PCM), 其差分版本称为 DPCM, 自适应版本则称为 ADPCM。考虑语音性质的变体也遵循以上定义。

在本章中, 我们将介绍自适应差分脉冲编码调制 (ADPCM)、声音合成器和更为通用的语音压缩: LPC、CELP、MBE 和 MELP。

### 13.1 语音编码中的 ADPCM

#### ADPCM

ADPCM 构成 ITU 的语音压缩标准 G.721、G.723、G.726 和 G.727 的核心 (参见 13.4 节中这些标准的代码)。这些标准的区别涉及码率和算法的某些细节。默认的输入是 $\mu$ 率编码的 PCM 16 位采样。ADPCM 的语音性能可以达到让 32kbps 的语音质量仅次于标准的 64kbps PCM 传输, 而优于 DPCM。

图 13-1 显示了说单词 “audio” 时产生的一秒的语音采样。在图 13-1a 中, 语音信号被保存为线性 PCM (和默认的 $\mu$ 律 PCM 相对), 每秒 8 000 个采样且每个采样为 16 位。在使用 G.721 进行 ADPCM 压缩后, 信号如图 13-1b 所示。图 13-1c 显示了实际和重建压缩信号之间的差值。尽管从技术角度来说, 两者的区别很明显, 但从感知的角度来说, 原始信号和压缩信号是非常相似的。

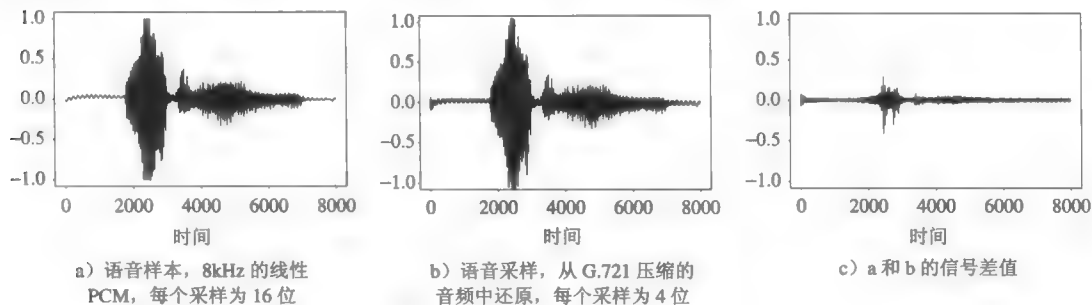


图 13-1 单词 “audio” 的波形

## 13.2 G.726 ADPCM

ITU 的 G.726 标准取代了 G.721 和 G.723。它提供了另一个版本的 G.711, 其中包括低码率的扩展。G.726 可以将 13 位或 14 位的 PCM 采样、8 位的 A 率或  $\mu$  率编码数据编码为 2~5 位的码字。它可以用于在数字网络、视频会议和 ISDN 通信中进行语音传输。

G.726 标准以一种简单的方式通过调整固定的量化器来工作。不同长度的码字使用 16kbps、24kbps、32kbps 或 40kbps 的码率, 其中采样率为 8kHz。标准定义了一个常量乘数  $\alpha$ , 它可以根据当前信号的尺度来改变差值  $e_n$ 。我们按如下方式定义比例化的差值信号  $f_n$ :

$$\begin{aligned} e_n &= s_n - \hat{s}_n \\ f_n &= e_n / \alpha \end{aligned} \quad (13.1)$$

这里  $\hat{s}_n$  表示预测信号值。  $f_n$  将被输入量化器进行量化。量化器如图 13-2 所示。这里, 输入值定义为差值和因子  $\alpha$  之间的比。

通过改变  $\alpha$  的值, 量化器可以在差值信号的范围内进行自适应改变。由于量化器是非均匀的中宽量化器, 所以它包括零值。量化器是后向自适应的。

原则上, 后向自适应的量化器会关心是否有太多的值被量化为远离零的值 (量化器的步长  $f$  太小) 或被量化为太过靠近零的区间 (量化器的步长太大)。

事实上, Jayant[1] 的算法允许我们在得到一次输出的结果后就调整后向量化器的步长。如果量化输入超出了量化范围, Jayant 量化器将增大步长, 反之则缩小步长。

假设我们有均匀量化器, 我们比较输入值的值域大小均为  $\Delta$ 。例如, 对于 3 位的量化器来说, 一共有  $k=0\sim7$  层。对于 3 位的 G.726, 则只使用 7 个层, 并在零的附近聚集。

Jayant 量化器为每一层赋予乘数值  $M_k$ , 靠近零的层的值小于 1, 而外层的值则大于 1。乘数值将与下一个信号值的步长相乘。这样, 外层值将会扩大步长, 很可能会使下一个量化值回到可用层的中间位置。而位于中间层次的量化值很可能会减小步长, 并且会使下一个量化值更加接近外层。

所以, 对于信号  $f_n$ , 量化器步长  $\Delta$  将会根据下面的简单公式, 随着前一个信号值  $f_{n-1}$  的量化值  $k$  变化:

$$\Delta \leftarrow M_k \Delta \quad (13.2)$$

由于信号的不同量化版本导致了这种变化, 这就可以认为是后向自适应的量化器。

在 G.726 中,  $\alpha$  如何变化取决于音频信号是真实的语音还是仅仅利用声带的的数据。在前一种情况下, 采样间的差值可能会有很大的波动, 而在后一种情况下则略有不同。为了适应不同的情况, 因子  $\alpha$  将根据一个由两部分组成的公式进行调整。

G.726 是使用固定量化步长的后向自适应 Jayant 量化器, 其步长是基于输入差值信号  $e_n$  除以  $\alpha$  的对数确定的。除数  $\alpha$  的对数形式为:

$$\beta \equiv \log_2 \alpha \quad (13.3)$$

由于我们希望区分差值较小和差值较大的情况, 所以  $\alpha$  被划分为锁定部分  $\alpha_L$  和非锁定部分  $\alpha_U$ 。我们的想法是, 把锁定部分作为小差值的比例因子, 它的变化非常缓慢; 而非锁定部分则用于大的差值。它们分别对应于对数  $\beta_L$  和  $\beta_U$ 。

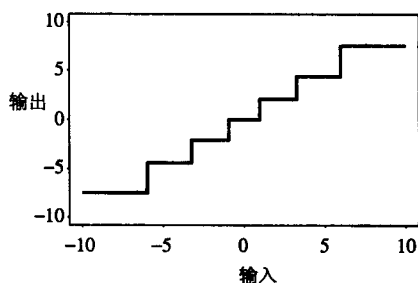


图 13-2 G.726 量化器

375  
376

对数值可以写成两部分值的和, 如下所示:

$$\beta = A\beta_U + (1-A)\beta_L \quad (13.4)$$

这里  $A$  可以变化, 对于语音它的值约为 1, 而对于声带数据, 它的值则约为 0。它可以根据信号的方差计算得到, 以反映过去的信号值。

非锁定部分根据下面的公式进行变化:

$$\begin{aligned} \alpha_U &\leftarrow M_k \alpha_U \\ \beta_U &\leftarrow \log_2 M_k + \beta_U \end{aligned} \quad (13.5)$$

其中,  $M_k$  是第  $k$  层的 Jayant 乘数。锁定部分通过对非锁定部分稍做修改得到, 修改的公式如下:

$$\beta_L \leftarrow (1-B)\beta_L + B\beta_U \quad (13.6)$$

这里  $B$  是一个很小的数, 例如  $2^{-6}$ 。

G.726 的预测程序比较复杂: 它使用 6 个量化差值和从先前 6 个信号值  $f_n$  中得到的两个重建信号值的线性合成。

### 13.3 声音合成器

到现在为止, 我们所研究的编码器(编码/解码算法)可以应用于任何类型的信号, 而不限于语音。声音合成器是专门的声音编码器。因此, 它无法应用于其他类型的模拟信号, 如调制解调器信号。

声音合成器考虑对语音建模, 以使用尽可能少的位捕获最为显著的特征。它们可以使用语音波形时间的模型(线性预测编码(Linear Predictive Coding, LPC)声音合成), 也可以将信号分解为频率分量后再进行建模(通道声音合成器和共振峰声音合成器)。

巧合的是, 我们都知道声音合成器对声音进行模拟的效果并不理想——当图书馆通过电话进行逾期通知时, 自动合成的语音会因为缺少语调而让人感觉非常怪异。

#### 13.3.1 相位不敏感性

回顾 8.5 节, 我们使用傅里叶分析的变形来进行分析, 从而将信号分解为其构成频率。原则上, 我们也可以使用于上述方法得到的频率系数来还原信号。但事实上, 从感知的角度来说, 将语音波形完全还原是没有必要的, 也就是说, 只要保证在任意时刻的能量基本正确, 那么信号听起来基本正确就可以了。

“相位”是在时间函数内时间参数的一个偏移量。假设我们按下一个钢琴键, 产生了类似于正弦曲线的声音  $\cos(\omega t)$ , 其中  $\omega=2\pi f$ 。如果我们等待足够长的时间以产生一个相位偏移  $\pi/2$ , 然后按下另外一个键, 产生声音  $\cos(2\omega t + \pi/2)$ , 就得到了如图 13-3 中实线所示的波形。这个波形是  $\cos(\omega t) + \cos(2\omega t + \pi/2)$  的和。

如果我们没有等待而是直接按下第二个音符 (1/4ms, 在图 13-3 中), 那么得到的波形将会是  $\cos(\omega t) + \cos(2\omega t)$ 。然而从感知上来说, 这两个音符听起来是相同的, 尽管它们之间存在着相位的偏移。

因此, 只要我们可以使得能量频谱分布正确(它决定我们听见的声音是响亮的还是轻微的), 那我们就无须担心波形的确切形式。

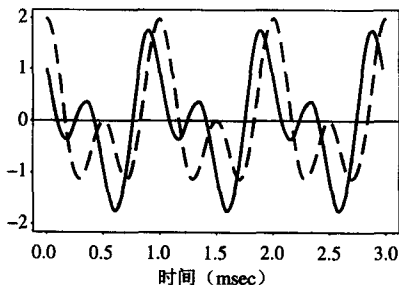


图 13-3 实线表示两段具有相位差的余弦曲线的叠加, 虚线表示没有相位差的叠加。两者的波形是不同的, 但听起来的效果却非常相似

### 13.3.2 通道声音合成器

子带滤波是使用让模拟信号通过一排带通滤波器的过程，以便实现傅里叶分析中的频率分解。子带编码是使用这个滤波过程所得到的信息来获得更好的压缩效果的过程。

例如，一个较早的 ITU 标准 G.722 使用子带滤波将模拟信号分解为两个频带：50Hz~3.5kHz 和 3.5~7kHz 的声音频率。然后，低频信号通过 48kbps 发送，我们可以很容易地听出差异；而高频信号则通过 16kbps 发送。

378

声音合成器可以在较低的码率下（1~2kbps）工作。为此，通道声音合成器首先使用一组滤波器将信号分解为不同的频率分量，如图 13-4 所示。然而，正如我们刚才所介绍的那样，只有能量是重要的，所以首先将波形“修正”为它的绝对值。这组滤波器将为每个频率范围产生能量的相对强度。子带编码器不会对信号进行修正，并将使用更宽的频带。

通道声音合成器还进行信号分析，以决定语音总的音调——低（男低音）或高（男高音），以及语音的激励。语音激励主要由声音是由浊音还是清音发出的。若一个声音是由清辅音发出的，其信号很简单，看起来像噪声；s 和 f 就发清音。而元音 a、e 和 o 就发浊音，其声波是周期性的。图 13-1 中单词“audio”结尾的 o 就有明显的周期性。在一个元音中，空气从声带以均匀而短促的气流压出，速率是男性 75~150 脉冲/秒，女性为 150~250 脉冲/秒。

辅音可以分为清辅音和浊辅音。而对于字母 m 和 n 的鼻音，声带的振动使声音从鼻腔中发出，而不是口腔。这些辅音是浊辅音。而对于 b、d 和 g，它们也是浊辅音，开始发音时嘴是闭合的，然后根据后面的元音打开，这个过程将经历几毫秒。浊辅音的能量高于清辅音，但低于元音。清辅音的例子包括 sh、th 和 h（当它们用于单词开头的时候）。

通道声音合成器应用声道转换模型来产生一个描述声音模型的激励参数向量。声音合成器同样也会猜测声音是浊音还是清音，对于浊音，还要估计其周期（也就是声音的音调）。图 13-4 显示解码器同样使用声道模型。

379

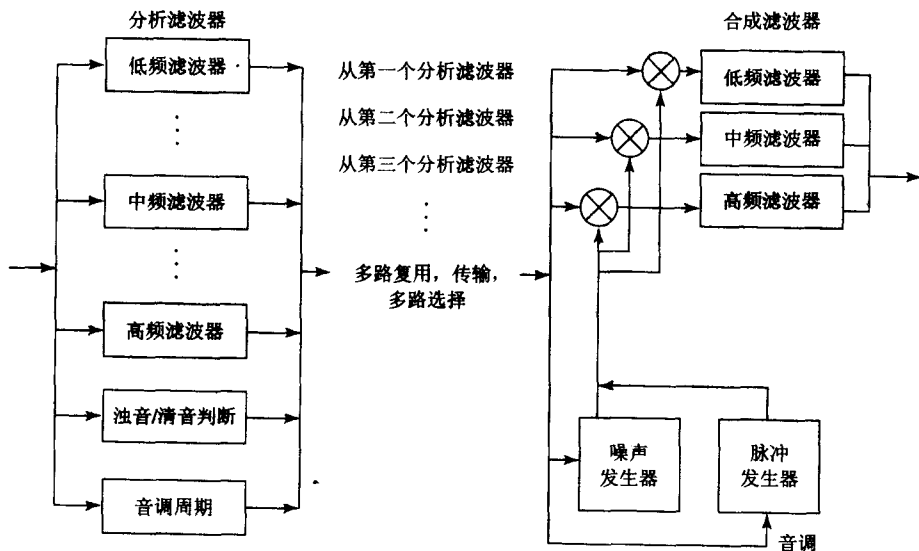


图 13-4 通道声音合成器

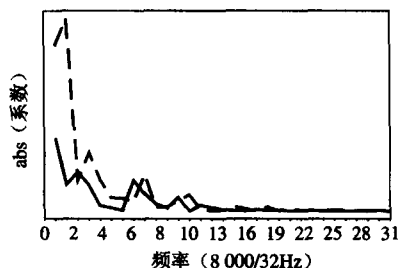
由于浊音可以通过正弦曲线来近似，一个周期性的脉冲发生器可以用来重现浊音。由于清音和噪声比较类似，因此一个伪噪声发生器可以用来重现清音，所有的值都通过由带通滤波器组得

到的能量估值来进行比例协调。通道声音合成器可以使用 2 400bps 得到一个智能的、但是是人工合成的语音。

### 13.3.3 共振峰声音合成器

在语音中，并非所有的频率都是等强度分布的，只有某个部分的频率非常强烈，其他部分的频率的则相对较弱。这是语音形成方式的直接结果，它是通过在嘴、喉、鼻等器官中的一些器室中共振得到的。重要的频峰称为共振峰（formant）[2]。

图 13-5 显示了上述结论是如何出现的：只有几个，通常为 4 个左右，处在某个频率的能量峰值。然而，随着语音的继续，峰值的出现点也随之变化。例如，两个不同的元音将会激活不同的共振峰——这反映了构成每个元音的不同的声道配置方式。通常，只需要分析一小段语音，如 10~40 毫秒，就可以找到共振峰。共振峰合成器通过对最重要的频率进行编码来工作，共振峰只能在 1 000bps 生成可接受的智能语音。



### 13.3.4 线性预测编码

LPC (Linear Predictive Coding, 线性预测编码)

声音合成器从波形中提取语音的最显著的特征，而不是将信号转化至频域。LPC 编码使用由给定激励产生的时变声道模型。传输的内容不是实际的信号或差值，而是用来对声道的形状和激励进行建模的参数。由于发送的是对声音的分析而非声音本身，因此使用 LPC 的码率很低。这和使用简单的描述符（如 MIDI）来生成音乐类似：我们仅仅发送描述参数，而让声音发生器尽可能地生成合适的音乐。不同之处在于除了音调、时长和响度变量之外，在这里我们还要发送声道激励参数。

在对一组称为片段（segment）或帧（frame）的数字化样本进行分析之后，由输出声道模型生成的语音信号将由当前语音输出加上先前模型系数的线性值的一个函数计算得到。这就是编码器名称中“线性”一词的由来。这个模型是自适应的，即编码器端会为每个新片段发送一组新的系数。

先前系数组的典型数量是  $N=10$ （“模型阶数”是 10），这样一个 LPC-10[3] 系统通常使用 2.4kbps 的速率。模型的系数  $a_i$  作为预测系数，并且要乘上先前的语音输出样本值。

LPC 首先决定当前片段是清音还是浊音。对于清音部分，使用宽带噪声发生器来生成样本值  $f(n)$  作为声道模拟器的输入。对于浊音部分，使用脉冲串发生器来生成值  $f(n)$ 。模型参数  $a_i$  则通过最小二乘方程组来计算，它将使实际语音和声道模型所产生的结果之间的差值最小。声道模型则通过捕获语音参数的噪声或脉冲串发生器来激励。

如果用  $s(n)$  表示产生的输出值，那么对于输入值  $f(n)$ ，输出取决于先前的输出样本值  $p$ ，它们之间有以下关系：

$$s(n) = \sum_{i=1}^p a_i s(n-i) + Gf(n) \quad (13.7)$$

这里， $G$  表示增益系数。注意，系数  $a_i$  作为线性预测器模型中的值。伪噪声发生器和脉冲发生器如上面的讨论，图 13-4 关于通道声音合成器的内容中有相关描述。

语音编码器以逐块编码的方式工作。输入的数字信号被划分成定长的小段进行分析，每个小段称为语音帧。对于 LPC 语音编码器，常用 22.5 毫秒的帧，即在 8kHz 采样的语音中有 180 个采



样点。语音编码器分析语音帧来获得各种参数,例如 LP 系数  $a_i (i=1 \dots p)$ 、增益  $G$ 、音调  $P$ 、以及清音/浊音判定  $U/V$ 。

为了求得 LP 系数,我们需要求解下列关于  $a_j$  的极小值问题:

$$\min E\{[s(n) - \sum_{j=1}^p a_j s(n-j)]^2\} \quad (13.8)$$

取  $a_i$  的导数,并设其为 0,我们得到一个由  $p$  个方程组成的方程组:

$$E\{[s(n) - \sum_{j=1}^p a_j s(n-j)]s(n-i)\} = 0, \quad i=1 \dots p \quad (13.9)$$

设  $\phi(i, j) = E\{s(n-i)s(n-j)\}$ , 我们有

$$\begin{bmatrix} \phi(1,1) & \phi(1,2) & \cdots & \phi(1,p) \\ \phi(2,1) & \phi(2,2) & \cdots & \phi(2,p) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(p,1) & \phi(p,2) & \cdots & \phi(p,p) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} \phi(0,1) \\ \phi(0,2) \\ \vdots \\ \phi(0,p) \end{bmatrix} \quad (13.10)$$

常常用自相关法来求解 LP 系数,其中

$$\phi(i, j) = \sum_{n=p}^{N-1} s_w(n-i)s_w(n-j) / \sum_{n=p}^{N-1} s_w^2(n) \quad i=0 \dots p, j=1 \dots p \quad (13.11)$$

$s_w(n) = s(n+m)w(n)$  是从时间  $m$  开始的加窗语音帧。因为  $\phi(i, j)$  仅仅由  $|i-j|$  决定,我们定义  $\phi(i-j) = R(|i-j|)$ 。因为  $R(0) \geq 0$ , 所以矩阵  $\{\phi(i, j)\}$  是正对称的,于是得到下面所示的快速计算 LP 系数的方法。

#### 过程 13-1 LPC 系数

```

E(0) = R(0), i = 1
while i ≤ p
    k_i = [R(i) - ∑_{j=1}^{i-1} a_j^{i-1} R(i-j)] / E(i-1)
    a_i^i = k_i
    for j = 1 to i-1
        a_j^i = a_j^{i-1} - k_i a_{i-j}^{i-1}
    E(i) = (1 - k_i^2) E(i-1)
    i ← i + 1
for j = 1 to p
    a_j = a_j^p

```

382

求得 LP 系数以后,用下面的公式来求增益  $G$ :

$$\begin{aligned} G &= E\{[s(n) - \sum_{j=1}^p a_j s(n-j)]^2\} \\ &= E\{[s(n) - \sum_{j=1}^p a_j s(n-j)]s(n)\} \\ &= \phi(0,0) - \sum_{j=1}^p a_j \phi(0,j) \end{aligned} \quad (13.12)$$

在自相关方案中,  $G = R(0) - \sum_{j=1}^P a_j R(j)$ 。阶数为 10 的信号分析完全能够满足语音编码的需要。

通过自相关法找到语音峰值的下标能够求出当前语音帧的音调  $P$ :

$$v(i) = \frac{\sum_{n=m}^{N-1+m} s(n)s(n-i)}{[\sum_{n=m}^{N-1+m} s^2(n) \sum_{n=m}^{N-1+m} s^2(n-i)]^{1/2}} \quad (13.13)$$

$$i \in [P_{\min}, P_{\max}]$$

对于 8kHz 的语音采样, 查找范围  $[P_{\min}, P_{\max}]$  通常设成  $[12, 140]$ 。假设  $P$  是峰值延时。如果  $v(P)$  小于某个阈值, 则当前语音帧被认为是浊语音帧, 在接收端可以通过一个白噪声序列模拟来加以重构。否则, 就认为当前语音帧是清音的, 在重构阶段使用一个周期性的波形来模拟。在实际的 LPC 语音编码器中, 音调估计和 U/V 判定一般基于动态编程方案, 这样可以纠正经常出现的在单个语音帧中音调重叠或者分割的错误。

在 LPC-10 中, 每个语音段有 180 个采样, 即在 8kHz 的采样频率下以 22.5 毫秒为间隔采样。传输的语音参数主要是系数  $a_k$ ,  $G$  (增益因子), 清音/浊音标志 (1 位), 以及清音语音帧的音调周期。

### 13.3.5 CELP

CELP 是码激励线性预测的首字母简写 (Code Excited Linear Prediction), 有时也称为码本激励 (codebook excited)。它是一组更加复杂的编码器, 它试图通过更加复杂的激励描述机制来弥补简单的 LPC 模型在语音质量上的缺陷。它使用一个完整的激励向量集合 (一个码本) 来和真实的语音匹配。把最佳匹配项的序号发送给接收者。这样做所导致的复杂性使得数据率增加到 4 800~9 600bps。

在 CELP 中, 因为所有的语音段使用的是同一个模板码本中的同一个模板集, 所以最后的语音听起来比 LPC-10 编码器中使用双态激励机制得到的结果要好很多。CELP 所达到的声音质量可以满足音频会议的需要。

383

在会议系统中, 我们需要较低的数据率, 而且语音的感受质量也必须在一个可以接受的水平。

在 CELP 中, 使用两种预测方法来消除语音信号中的冗余, 它们是长时预测 (LTP) 和短时预测 (STP)。STP 是一种对采样点进行分析的技术, 它希望通过前面已有的预测值来推测下一个采样值。在这里, 产生冗余的主要原因是一个采样点相对前面几个采样点不会产生太大的变化。LTP 的主要思想是, 对于语音特别是浊音, 在一段内或者在段与段之间, 一个基本的周期或音调会使得同一个波形反复出现。我们可以通过找到语音音节的方法来消除这种冗余。

假设我们用 8kHz 来采样, 使用 10 毫秒的帧, 一共 80 个采样点。我们希望的大致的重复的长度是 12~140 个采样点 (实际上每个音节比我们实际选择的帧长要长)。

STP 基于短时 LPC 分析, 我们将在后面的小节中讨论。之所以称为“短时”, 是因为在预测过程中, 只有一些采样点, 而不涉及整个一帧或者几帧语音。STP 也要求整个语音帧中的残差最小, 不过它仅仅体现了很短的一段语音之间的关联 (在 10 阶 LPC 中大概是 10 个采样点)。

经过 STP 处理后, 我们用原信号值减去预测值, 就进入了差分编码的阶段。然而, 即使是在误差值集合  $e(n)$  中, 序列中还存在基本的语音音节。这时候就要使用 LTP。LTP 主要是用来进一步消除清音语音信号中的周期冗余。简单来讲, STP 能够提取短时语音频谱中的共振峰结构, 而 LTP 能够恢复表示语音音节的语音信号中的长期关联。

因此我们通常使用两个操作步骤, 首先是 STP, 然后是 LTP。因为始终假设开始的误差值为

0, 然后开始处理一个语音分量 (如果我们使用闭循环, 那么首先执行 STP)。接着使用 LTP 来处理整个语音帧, 或者相当于 1/4 帧的子语音帧。图 13-6 显示了这两个步骤。

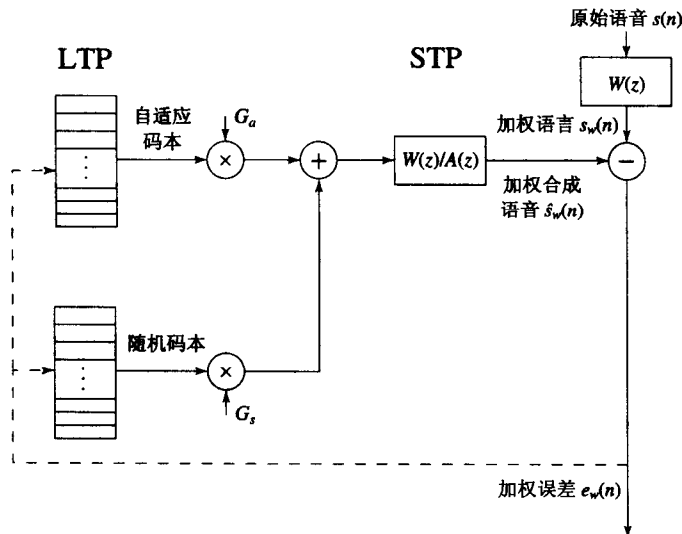


图 13-6 使用自适应随机码本的 CELP 分析模型

LTP 通常实现为自适应码本查找。自适应码本中的码字是一个移位语音残差段, 通过一个对应于当前语音帧或当前子帧的延时值  $\tau$  来标记它。其方法是在波形码本中寻找能和当前的语音帧匹配的码字。一般使用正规化的语音子帧在码本中寻找, 所以当语音段能够匹配的时候, 我们也获得了一个变换值 (增益)。和码字相对应的增益值标记成  $g_0$ 。

有两种码字检索方法: 开环和闭环。开环自适应码字查找试图使长程预测误差最小, 但不能使感知上的重建语音误差最小。

$$E(\tau) = \sum_{n=0}^{L-1} [s(n) - g_0 s(n-\tau)]^2 \quad (13.14)$$

我们把  $g_0$  的偏导数设置为 0, 即  $\partial E(\tau) / \partial g_0 = 0$ , 我们能够得到

$$g_0 = \frac{\sum_{n=0}^{L-1} s(n)s(n-\tau)}{\sum_{n=0}^{L-1} s^2(n-\tau)} \quad (13.15)$$

因此, 最小的求和误差是:

$$E_{\min}(\tau) = \sum_{n=0}^{L-1} s^2(n) - \frac{[\sum_{n=0}^{L-1} s(n)s(n-\tau)]^2}{\sum_{n=0}^{L-1} s^2(n-\tau)} \quad (13.16)$$

需要注意的是, 采样  $s(n-\tau)$  可能在前一语音帧中。

现在, 为了获得自适应码本的索引  $\tau$ , 我们可以在一个音节确定的一小段范围进行搜索。

更多的时候, CELP 编码器使用闭环搜索。与仅仅考虑平方和不同, 通过自适应的码本查找, 语音重构时所感觉到的误差最小。所以在闭环自适应码本查找中, 在自适应码本中查找中的最佳选择是使局部重构语音失真最小的码字。参数的选取是通过求源和重建语音之间的差值度量 (通常是均方) 最小化得到的。这意味着我们在分析语音段的同时也在做语音的合成, 所以这种方法

也称为合成分析法 (Analysis-By-Synthesis, A-B-S)。

384  
385

经过基于 LPC 分析的 STP 变换和基于自适应码本查找的 LTP 变换后, 我们得到的残差信号有点类似于白噪声, 通过在一个码本中匹配的码字来编码, 各个码字出现的概率是随机的 (可以是完全任意的, 或者是遵循一定的概率分布)。这种对于自适应码字和随机码字的序列化的优化之所以被采用, 是因为相关的优化自适应的随机分布的码字太复杂, 不可能满足实时的要求。

解码就是上面所描述过程的逆过程, 把两种激励机制的贡献结合在一起进行工作。

#### 1. DOD 4.8 KBPS CELP (FS1016) \*

DOD 4.8 kbps CELP[4]是被美国联邦标准委员会采用的早期的 CELP 算法, 它被当作是对 2.4 kbps LPC-10e (FS1015) 声音编码器的升级。这种声音编码器现在是测试其他低码率声音编码器的基准。FS1016 的采样率为 8kHz, 每帧有 30 毫秒。而且每帧被进一步划分为 7.5 毫秒的子帧。在 FS1016 中, STP 基于 10 阶的开环 LPC 分析。

为了提高编码效率, 使用一个复杂的变换编码。然后使用变换系数来进行量化和压缩。

首先, 人们一般使用  $z$  变换。这里  $z$  是一个复数, 代表一种复“频率”。如果  $z = e^{-2\pi i/N}$ , 那么离散  $z$  变换退化成离散傅里叶变换。 $z$  变换使得傅里叶变换看起来像是多项式。我们可以把预测方程中的误差写成:

$$e(n) = s(n) - \sum_{i=1}^p a_i s(n-i) \quad (13.17)$$

因为在  $z$  域, 有

$$E(z) = A(z)S(z) \quad (13.18)$$

其中  $E(z)$  是误差的  $Z$  变换,  $S(z)$  是信号的变换。 $A(z)$  是  $Z$  域的传递函数, 等价于

$$A(z) = 1 - \sum_{i=1}^p a_i z^{-i} \quad (13.19)$$

这里的系数  $a_i$  和式 13.7 中的系数  $a_i$  相同。我们重构语音时根据下式完成:

$$S(z) = E(z)/A(z) \quad (13.20)$$

因此, 一般  $A(z)$  一般都写成  $1/A(z)$ 。

使用  $Z$  变换的主要原因是想把 LP 系数变换成线谱对 (LSP) 系数。在 LSP 空间中, 量化操作有一些比较好的性质。LSP 表示法现在已经成为标准, 并且应用到最近的几乎所有基于 LPC 的语音编码器中, 例如 G.723.1、G.729 以及 MELP。为了获得 LSP 系数, 我们构建两个多项式

$$\begin{aligned} P(z) &= A(z) + z^{-(p+1)} A(z^{-1}) \\ Q(z) &= A(z) - z^{-(p+1)} A(z^{-1}) \end{aligned} \quad (13.21)$$

其中  $p$  是 LPC 分析中的阶数,  $A(z)$  是 LP 滤波器的转换函数,  $z$  是转换域的变量。 $Z$  变换和傅里叶变换非常相似, 只不过它使用的是复“频率”。

386

上面两个多项式的根分布在  $z$  平面的单位圆附近, 而且关于  $x$  轴对称。我们把  $P(z)$ 、 $Q(z)$  分布在  $x$  轴上方的根的相位角记成  $\theta_1 < \theta_2 < \dots < \theta_{p/2}$  以及  $\varphi_1 < \varphi_2 < \dots < \varphi_{p/2}$ , 其中  $p$  是偶数。那么向量  $\{\cos(\theta_1), \cos(\varphi_1), \cos(\theta_2), \cos(\varphi_2), \dots, \cos(\theta_{p/2}), \cos(\varphi_{p/2})\}$  就是 LSP 系数向量。向量  $\{\theta_1, \varphi_1, \theta_2, \varphi_2, \dots, \theta_{p/2}, \varphi_{p/2}\}$  通常被称作线谱频率 (Line Spectrum Frequency) 或者 LSF。因为  $A(z) = [P(z) + Q(z)]/2$ , 我们能够在解码端根据 LSP 或者 LSP 系数重构 LP 系数。

FS1016 中的自适应码本查询是采用基于感受误差的闭环查找, 和仅仅考虑均方差不同, 这里

的误差考虑了人的感受能力。根据 Z 变换, 可以发现下列乘法器能够发挥很好的作用:

$$W(z) = \frac{A(z)}{A(z/\gamma)} = \frac{1 - \sum_{i=1}^p a_i z^{-i}}{1 - \sum_{i=1}^p a_i \gamma^i z^{-i}} \quad 0 < \gamma < 1 \quad (13.22)$$

其中参数  $\gamma$  是常数。

自适应码本中含有 256 个码字, 其中 128 个是整数时延, 128 个是非整数时延 (带有 1/2 个采样值步长, 以提供更好的分辨率)。整数时延的范围是 20~147。为了降低查找复杂度, 偶子帧在和前一个奇子帧相隔 1 个步长的范围内查找。差分使用 6 位来编码。增益使用非均匀编码, 可能使用 1、2 或者 5 位。

对一帧中的 4 个子帧都使用随机码本 (Stochastic Codebook) 检索。把一个单位方差高斯分布随机序列的阈值限制在绝对值 1.2 内, 然后把序列中的元素量化为 -1, 0, 1, 这样就生成了 FS1016 中的随机码本。随机码本中有 512 个码字。码字之间是互相重叠的, 每一个码字比前一个码字偏移 2。这种随机设计称为代数码本。它有多种变体, 广泛用于最近的 CELP 编码器中。

把激励向量记成  $v^{(i)}$ , 我们在第一步获得周期分量  $v^{(0)}$ 。 $v^{(1)}$  是第二步随机变量查找的结果。在闭环查找中, 重构语音可以写成

$$\hat{s} = \hat{s}_0 + (u + v^{(i)})H \quad (13.23)$$

其中第一步中  $u$  为 0, 在第二步中设置  $v^{(0)}$ 。 $\hat{s}_0$  是 LPC 重构滤波器的零响应。矩阵  $H$  是简化的 LPC 重构滤波器单位脉冲响应矩阵。

$$H = \begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_{L-1} \\ 0 & h_0 & h_1 & \cdots & h_{L-2} \\ 0 & 0 & h_0 & \cdots & h_{L-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & h_0 \end{bmatrix} \quad (13.24)$$

387

其中  $L$  是子帧的长度 (在这里代表了一个卷积)。类似地, 定义  $W$  为感应权重滤波器的单位响应矩阵, 那么重构语音的感应权重误差是

$$e = (s - \hat{s})W = e_0 - v^{(i)}HW \quad (13.25)$$

其中  $e_0 = (s - \hat{s}_0)W - uHW$ 。码本搜索过程是要在码本中找到一个码字  $y^{(i)}$  以及对应的  $a^{(i)}$ , 使得  $v^{(i)} = a^{(i)}y^{(i)}$ , 并且  $ee^T$  取最小值。为了使问题易于处理, 顺序检索自适应码本和随机码本。我们用  $\tilde{a}^{(i)} = Q[\hat{a}^{(i)}]$  代表量化后的值, 这样自适应码本或随机码本中码字检索的判断标准是使得对所有的  $y^{(i)}$ 、 $ee^T$  取得最小值。其中  $y^{(i)}$  是关于  $\tilde{a}^{(i)}$ 、 $e_0$ 、 $y^{(i)}$  的函数。

CELP 的解码器完成上述过程的逆过程。因为向量量化的复杂性是非对称的, 解码端的复杂性要低好多。

## 2. G.723.1\*

G723.1[5] 是 ITU 颁布的标准, 主要目的是为了实现多媒体通信。它能够和 H.324 协议结合, 从而在视频会议系统中实现音频编码。G.723.1 是一个双数据率的 CELP 类型的语音编码器, 它可以数据率 5.3kbps 和 6.3kbps 下工作。

G723.1 中使用很多和 FS1016 类似的技术, 我们将在最后一节详细讨论。输入的语音也是 8kHz, 以 16 位的线性 LPC 格式采样。每帧是 30 毫秒, 也被划分成四个相同大小的子帧。每个

子帧中采用 10 阶的 LPC 系数。LP 系数进一步转换成 LSP 向量,通过预测划分 VQ 算法进行量化。LP 系数也用来生成感应权重滤波器。

G.723.1 首先使用开环音节划分器 (estimator) 来进行一次粗糙的音节划分。每个音节的长度为两个子帧。在每个子音节中都进行闭环音节查找,方法是在开环音节范围中寻找所要的数据。在经过 LP 滤波并且通过 LTP 消除语音中的谐波成分后,如果是 5.3kbps 的编码器,那么通过 MP-MLQ (Multipulse Maximum Likelihood Quantization) 对随机残差进行量化,如果是 6.3kbps 的编码器,那么通过 ACELP (Algebraic Code Excited Linear Prediction) 来对随机残差进行量化,这种方法能获得更好的语音质量。这两种量化方法可以在不同的长度为 30 毫秒的语音帧之间切换。

在 MP-MLQ 中,随机分量的贡献体现为一串脉冲:

$$v(n) = \sum_{i=1}^M g_i \delta(n - m_i) \quad (13.26)$$

其中  $M$  是脉冲的数目,  $g_i$  是脉冲在位置  $m_i$  处的增益。闭环查找要使得下式取得最小值:

$$e(n) = r(n) - \sum_{i=1}^M g_i h(n - m_i) \quad (13.27)$$

其中  $r(n)$  是感应估计后消除了零响应分量和周期分量后余下的语音成分。使用在上一节中类似的方法,我们可以逐步优化每一个脉冲的增益和位置。也就是说,我们首先假设只有一个脉冲,并且找到最佳的位置和增益。消除了这个脉冲的影响后,我们可以用同样的方法来优化下一个脉冲。这个操作可以递归地执行,直到我们优化了所有的  $M$  个脉冲为止。

ACELP 中随机码本的结构和 FS1016 中码本的结构不同。下表所示的就是一个 ACELP 激励码本。

符号	位置	
$\pm 1$	0, 8, 16, 24, 32, 40, 48, 56	
$\pm 1$	2, 10, 18, 26, 34, 42, 50, 58	
$\pm 1$	4, 12, 20, 28, 36, 44, 52, (60)	(13.28)
$\pm 1$	6, 14, 22, 30, 38, 46, 54, (62)	

一共有 4 个脉冲,每个脉冲可以有 8 个位置,每个位置用 3 位进行编码。脉冲的符号占 1 位,另一个位用于把所有可能位置平移,使之成为奇数。这样,码字的索引需要 17 位。因为代数码本的特殊结构,有一种高效的码字查找算法。

除了上面讨论的 CELP 编码外,还有一些其他的 CELP 类的编解码方法,它们主要用于无线通信系统。这些编码器的基本原理非常类似,只是在参数分析以及码本结构的实现细节上有一些不同。

相关的例子有数据率为 12.2kbps 的 GSM EFR (Enhanced Full Rate) 代数 CELP 编解码器[6],以及专门为北美的数字电话网 IS-136 TDMA 系统设计的 IS-641EFR[7]。G.728[8]也是一种低延时的 CELP 语音编码器。G.729[9]是基于 ITU 标准的 CELP,目标是收费质量的语音通信。

G.729 是一个 CS-ACELP (Conjugate-Structure Algebraic-Code-Excited-Linear-Prediction) 编解码标准。G.729 的语音分析帧的长度为 10 毫秒,所以比使用 30 毫秒语音帧的 G.723.1 延时小。G.729 还设置了一些保护措施来处理应用 (例如 VoIP) 中的丢包问题。

### \*13.3.6 混合激励声音合成器

混合激励声音合成器是另外一类语音编码标准。它们和 CELP 不同,在 CELP 中,激励体现

为自适应码字和随机码字。与此相反,混合激励编码器使用基于模型的方法引进了多模型激励的概念。

### 1. MBE

多频带激励(Multi-Band Excitation, MBE)声音合成器是由MIT的林肯实验室开发的。4.15kbps的IMBE编解码器[11]是IMMSAT的标准。MBE是一种逐块编解码标准,用长度为20~30毫秒的语音帧作分析单元。在MBE解码器的分析阶段,首先对当前帧中加窗语音应用频谱分析,如一个FFT变换。短时语音频谱被进一步划分成更小的频段。带宽通常是基频(等于音节的倒数)的整数倍,每个频段都被标记为“浊音”或者“清音”。

389

MBE参数编码器因此包括谱包络、音调、不同频带的浊音/清音(U/V)判断。基于不同码率的要求,谱的相位可以被参数化或被丢弃。在语音解码的过程中,浊音频带和清音频带由不同的方法合成,生成最后的输出。

在参数估计阶段,MBE使用合成分析技术。一些参数,例如基频、频谱包络,以及子带U/V决策都是通过闭环查找得到的。闭环优化的评价原则是使考虑感受权重重的重构语音误差最小。重构语音误差在频域的表达式是:

$$\varepsilon = \frac{1}{2\pi} \int_{-\pi}^{+\pi} G(\omega) |S_w(\omega) - S_{wr}(\omega)| d\omega \quad (13.29)$$

其中 $S_w(\omega)$ 是原始语音的短时频谱, $S_{wr}(\omega)$ 是重构语音的短时频谱, $G(\omega)$ 是感应权重滤波器的频谱。

和CELP中的闭环查找类似,使用一个序列化的优化方法可以使问题变得简单。首先假设所有频段都是清音的,然后计算频谱的包络以及基频。根据所有的频谱的都是清音的假设来重新计算频带误差,得到公式

$$\tilde{\varepsilon} = \sum_{m=-M}^M \left[ \frac{1}{2\pi} \int_{\alpha_m}^{\beta_m} G(\omega) |S_w(\omega) - A_m E_{wr}(\omega)|^2 d\omega \right] \quad (13.30)$$

其中 $M$ 是在 $[0, \pi]$ 中的一个频带号, $A_m$ 是频带 $m$ 的频谱包络, $E_{wr}(\omega)$ 是短时窗口频谱, $\alpha_m = (m - \frac{1}{2})\omega_0$ ,  $\beta_m = (m + \frac{1}{2})\omega_0$ 。我们设 $\partial \tilde{\varepsilon} / \partial A_m = 0$ , 则有

$$A_m = \frac{\int_{\alpha_m}^{\beta_m} G(\omega) S_w(\omega) E_{wr}^*(\omega) d\omega}{\int_{\alpha_m}^{\beta_m} G(\omega) |E_{wr}(\omega)|^2 d\omega} \quad (13.31)$$

在频带间隔中查找最小 $\tilde{\varepsilon}$ 值时,同时也可以获得基频。根据求得的频谱包络,使用一个自适应的阈值机制来测试每个频段的匹配程度。如果一个频段匹配程度很好,我们就称这个频段是浊音的。否则我们就称这个频段是清音的,同时测试清音频段的包络。

$$A_m = \frac{\int_{\alpha_m}^{\beta_m} G(\omega) |S_w(\omega)| d\omega}{\int_{\alpha_m}^{\beta_m} G(\omega) d\omega} \quad (13.32)$$

解码器基于浊音和清音的频段,使用不同的方法合成浊音和清音语音。然后把两种重构分量组合在一起,生成合成语音。最后一步是处理不同语音帧之间的迭加部分,然后得到最终输出。

390

### 2. MELP

MELP (Multiband Excitation Linear Predictive, 多频带激励线性预测) 语音编解码是新颁布的

美国联邦标准, 用来取代旧的 LPC-10 (FS1015) 标准, 它主要应用于低数据率的安全通信。当数据率是 2.4kbps 时, MELP[12] 的声音质量可以与 4.8kbps 的 DOD-CELP (FS1016) 媲美, 同时 MELP 在噪声环境下有很好的健壮性。

MELP 也是基于 LPC 分析的。根据 LPC-10 中采用的硬性决策的浊音或者清音模型的不同, MELP 使用多频段软决定模型来处理激励信号。LP 残差是带通的, 对每个频段计算一个语音强度。把激励通过 LPC 合成滤波器传递就可以获得重构语音。

和 MBE 不同的是, MELP 把激励划分成五个固定频段, 它们是 0~500、500~1 000、1 000~2 000、2 000~3 000 以及 3 000~4 000Hz。使用语音信号的正则相关函数计算语音信号和平滑的非直流频段的校正信号来计算每个频带上的语音程度参数。假设用  $s_k(n)$  来代表频段  $k$  上的语音信号,  $u_k(n)$  代表  $s_k(n)$  的平滑的无直流分量的信号。相关函数可以写成

$$R_x(P) = \frac{\sum_{n=0}^{N-1} x(n)x(n+P)}{[\sum_{n=0}^{N-1} x^2(n) \sum_{n=0}^{N-1} x^2(n+P)]^{1/2}} \quad (13.33)$$

其中  $P$  是当前语音帧中的音节,  $N$  是帧长。那么频段  $k$  的语音强度可以定义成  $\max(R_{sk}(P), R_{uk}(P))$ 。

在清音语音段中, MELP 使用抖动有声状态来模拟清音语音段的边缘, 从而进一步取代了传统的 LPC-10 语音编码器中使用的方法。抖动状态使用一个非周期性的标志来标记。如果这个非周期性的标志出现在分析的末尾, 那么接收器在短时脉冲激励中增加一个随机移位分量。这种移位最大可以达到  $P/4$ 。抖动状态由全波峰值矫正的 LP 残差  $e(n)$  决定。

$$\text{peakiness} = \frac{[\frac{1}{N} \sum_{n=0}^{N-1} e(n)^2]^{1/2}}{\frac{1}{N} \sum_{n=0}^{N-1} |e(n)|} \quad (13.34)$$

如果峰值高于一些阈值, 那么就认为语音帧是抖动的。

为了更好地重构语音信号的短时频谱, 和在 LPC-10 语音编码器中一样, 我们假设残差信号的频谱不是均匀分布的。正则化 LP 残差信号后, MELP 保留了基频为  $\min(10, P/4)$  的谐波所对应的幅值。更高频的谐波被过滤掉。

10-d 幅值向量利用 8 位的向量来量化, 使用感应权重距离作为评价标准。和最流行的 LPC 量化方案类似, MELP 也是首先将 LPC 参数转变成 LSF, 然后使用四步向量量化方案。四个步骤中使用的位数分别是 7、6、6、6。除了使用和 LPC-10 中类似的整数音节估计外, MELP 还使用了一个分数音节精化过程, 来提高音节估计的准确度。

在音频的重构过程中, MELP 没有使用短时脉冲来模拟短时激励信号, 它使用离散的波形。为了使用离散的脉冲, 在脉冲上使用了一个有限的脉冲响应 (Finite Impulse Response, FIR) 滤波器。MELP 也使用感应权重滤波器对重构语音信号进行后滤波, 这样可以降低量化噪声, 提高语音质量。

### 13.4 进一步探索

在 Spania 的一篇很出色的文章[13]中有关于语音编码详细介绍。Sun Microsystem 公布了它对 G.711、G.721 和 G.723 标准的 C 代码实现。代码可以在本书网站的第 13 章的 Further Exploration 中找到, 同时可以找到样本音频文件。因为代码是处理 2 字节的原始数据的, 所以链接中也给出了简单的 MATLAB 程序来读写 WAV 以及原始数据。



对于音频来说,各种诉求的最终判决是由 ITU 标准来完成的。这些组织提出的标准使得我们的 Modem 能够和其他人交流,促进移动通信的开发等。负责制定和推广标准的 ITU 可以直接通过 Web 访问。

关于语音编码的更多的信息可以从语音 FAQ 文件链接中找到,其中还包含关于 LPC 以及 CELP 的实现代码。

### 13.5 练习

1. 我们在 13.3.1 节中讨论了相位不敏感性,请根据复合信号中单个频率分量解释术语“相位”的含义。
2. 通过 C 或者 MATLAB 来输入一个语音段,并证明共振峰的存在——一般任何语音段都有一部分重要的频率。同时证明随着我们检测的语音的时间长度的变化,共振峰也会随着改变。要编写一个频率分析器,最简单的做法是使用 8.5 节中 DCT 编码的思想。一维空间中的 DCT 变换可以写作:

$$F(u) = \sqrt{\frac{2}{N}} C(u) \sum_{i=0}^{N-1} \cos \frac{(2i+1)u\pi}{2N} f(i) \quad (13.35)$$

其中  $i, u = 0, 1, \dots, N-1$ , 其中  $C(u)$  是常数, 它的表达式是:

$$C(u) = \begin{cases} \frac{\sqrt{2}}{2} & u = 0 \\ 1 & \text{其他} \end{cases} \quad (13.36)$$

如果我们使用图 6-15 中的语音采样,对开始(或者最后的)40 毫秒语音(即 32 个采样)做一维 DCT 变换,我们就获得图 13-5 中的绝对频率分量。

3. 编写一段代码来读入一个 WAV 文件,你需要使用以下一些定义:一个 WAV 文件以 44 字节的文件头开始,数据类型是无符号字节。一些重要的参数信息编码如下:

字节[22~23] 通道数目

字节[24~27] 采样率

字节[34~35] 采样位数

字节[40~43] 数据长度

4. 编写一段程序为一段音频文件(WAV 格式)添加渐强以及减弱的效果。渐变效果的细节如下:算法中假设有一段线性包络;数据的最初 20%采样是渐强效果的作用范围,最后 20%的采样是渐弱效果的作用范围。

进一步的,你可以让你的代码不仅处理单声道 WAV 文件,还能够处理立体声 WAV 文件。如果需要的话,可以对输入文件的大小做一个限制,最大尺寸为 16MB。

5. 在本书中,我们研究 ADPCM 中的自适应量化机制,我们也使用自适应的预测机制。让我们来考察一个只有一步的预测,  $\hat{s}(n) = a \cdot s(n-1)$ 。写出应该怎样在开环方法中计算参数  $a$ 。计算你能够获得的 SNR 增益,并且和基于均匀量化的直接 PCM 方法作比较。
6. 线性预测分析能够检测短时频谱的包络的形状。已知 10 个 LP 系数  $a_1, \dots, a_{10}$ , 我们怎样求得共振峰的位置以及带宽?
7. 下载并且实现一个 CELP 编码器(参见本书的网站)。用你自己录制的声音数据测试这个编码器。
8. 在 G723.1 中量化 LSP 向量时,使用分裂向量量化。如果 LSP 是 10 维的,我们可以把这个向量分裂成 3 个子向量,分别是 3 维、3 维以及 4 维。然后对每个子向量分别使用向量量化。比

较一下使用和不使用分裂向量量化的码本空间的复杂性。计算使用了分裂向量量化后码本查询时间复杂度的提高。

9. 讨论一下在 CELP 中使用代数码本的优点。
10. 在背景噪声很强的情况下, LPC-10 语音编码器的编码质量会严重恶化, 讨论一下为什么 MELP 在同样的噪声环境下能够获得较好的效果。
11. 给出一个基于自相关函数的简单的估计音节的时域方法。如果这种方法仅仅基于语音帧, 会遇到什么问题? 如果我们有 3 个语音帧, 包括一个“过去帧”以及一个“将来帧”, 我们可以怎样改进估计结果?
12. 在接收端, 语音一般是使用两个语音帧参数, 而不是一个语音帧参数来生成, 这样可以避免突变。给出两种能够获得平滑转换的方法。使用 LPC 编解码来说明你的想法。

### 13.6 参考文献

- [1] N.S. Jayant and P. Noll, *Digital Coding of Waveforms*, Englewood Cliffs, NJ: Prentice-Hall, 1984.
- [2] J.C. Bellamy, *Digital Telephony*, New York: Wiley, 2000.
- [3] Thomas E. Tremain, "The Government Standard Linear Predictive Coding Algorithm: LPC-10," *Speech Technology*, April 1982.
- [4] J.P. Campbell, Jr., T.E. Tremain, and V.C. Welch, "The DOD 4.8 kbps Standard (Proposed Federal Standard 1016)," In *Advances in Speech Coding*. Boston: Kluwer Academic Publishers, 1991.
- [5] *Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 kbit/s*, ITU-T Recommendation G.723.1, March, 1996.
- [6] *GSM Enhanced Full Rate Speech Transcoding (GSM 06.60)*, ETSI Standards Documentation, EN 301 245, 1998.
- [7] *TDMA Cellular/PCS Radio Interface-Enhanced Full Rate Speech Codec*, TIA/EIA/IS-641 standard, 1996.
- [8] *Coding of Speech at 16 kbit/s Using Low-Delay Code Excited Linear Programming*, ITU-T Recommendation G.728, 1992.
- [9] *Coding of Speech at 8 kbit/s Using Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP)*, ITU-T Recommendation G.729, 1996.
- [10] D. W. Griffin and J. S. Lim "Multi-Band Excitation Vocoder," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(8): 1223-1235, 1988.
- [11] "Inmarsat [International Mobile Satellite]-M Voice Codec, v2," *Inmarsat-M Specification*, Feb 1991.
- [12] A.V. McCree and T.P. Barnwell, "Mixed Excitation LPC Vocoder Model for Low Bit Rate Speech Coding," *IEEE Transactions on Speech and Audio Processing*, 3(4): 242-250, July 1995.
- [13] A. Spanias, "Speech Coding: A Tutorial Review," *Proceedings of the IEEE*, 82: 1541-1582, 1994.

## 第 14 章 MPEG 音频压缩

你是否有过这样的经历：在参加一个舞会的时候，发现过了一段时间以后就听不到太多声音了？你正在经历一个典型的时间遮掩！

你是否注意到，一个在舞会负责控制音响的人，基本上听不到高频的声音？因为很多技师都有这样的听觉损伤，所以可以通过提高高频声音的音量来让他们能够听到这些高频声音。但是如果你的听觉没有这样的损伤，那么在听到这种音乐的时候会感觉太过刺耳。

另外，如果一个音色的声音非常大，那么你也会注意到很难听到这个频谱附近的其他音色——乐队主唱的声音会被主吉他手的声音所淹没。如果你已经注意到这种现象，那么你已经体会过了频率遮掩（frequency masking）！

MPEG 声音利用了这类感知现象，去掉了这些听不到的声音。利用人体的听觉敏感度曲线，MPEG 音频编码器决定在何时，以何种程度使用时间遮掩和频率遮掩让音乐的某些部分不被听到。然后，通过控制量化过程以使得这些部分不影响输出的效果。

在前面几章中，我们已经集中关注了电话上的应用——通常，LPC（线性预测编码）和 CELP（码激励线性预测）被调整到语音参数。不同的是，在本章中，我们考虑的压缩方法适用于一般的声音，比如音乐或者是广播数字电视。这种方法不使用语音建模的方法，而是使用一种波形（waveform）编码的方法，这种方法可以使解压缩后的幅值-时间波形和输入信号尽可能相似。

在评价可能的音频压缩方式的声音满意程度时使用的主要技术是利用一种心理声学听觉的建模方式。这种编码过程通常称为认知编码（perceptual coding）。

在本章中，我们会看到这些因素是如何影响 MPEG 音频压缩标准，并且详细介绍以下主题：

- 心理声学
- MPEG-1 音频压缩
- MPEG 音频的发展：MPEG-2、4、7 和 21

### 14.1 心理声学简介

前面提到过，人类的听觉范围是从 20Hz 到大约 20kHz（对于那些没有去过舞厅的人来说）。频率更高的声音称为超声波（ultrasonic）。但是，人类声音的频率范围只是从大约 500Hz 到 4kHz。这个动态范围（人类能够听到的最大声音幅度和最小声音幅度之比）在大约 120dB 这个数量级上。

回想一下，分贝单位是在指数轴上的强度之比。0dB 代表人类听觉的阈值——我们可以听见的最小的声音，此时的声音频率是 1kHz。从技术上说，这是一个刚能听到的声音，强度为每平方米  $10^{-12}$  瓦特。我们对强度感知的范围则是不可思议的大，使人耳感到疼痛的声音的强度大约是  $1\text{Watt/m}^2$ ，所以两者之间的此值可达到  $10^{12}$ 。

听觉范围事实上是和频率有关的。在频率为 2kHz 的时候，耳朵可以轻松地对比最小可感知声音频率大约强 96dB 的声音做出反应，换句话说是对  $2^{32}$  比率的激励做出反应。表 6-1 列出了一些常见声音对应的分贝数。

#### 14.1.1 等响度关系

假设我们产生两个纯音色（正弦声音波），它们有相同的幅值，但频率不同。通常，一个声音会比另外一个声音听起来更响。这是因为耳朵听低频或者高频的时候不能够像听到中频率声音

那样好。特别是，在通常的音量下，耳朵对 1kHz 到 5kHz 的频率最敏感。

### Fletcher-Munson 曲线

Fletcher-Munson 等响度曲线显示了一个给定的音量（声压值，以 dB 为单位）和感知的响度（以方为单位）之间的关系，它是一个关于频率的函数。图 14-1 显示了耳朵对等响度的感知特点。横坐标（显示在半指数图中）是频率，以 kHz 为单位。纵坐标是声压值——在实验中产生的真实的音色响度。这条曲线显示了何种音色响度能够被人类所接收。最下面的一条曲线显示何种模拟纯音色的值能够产生感知是 10dB 的声音。

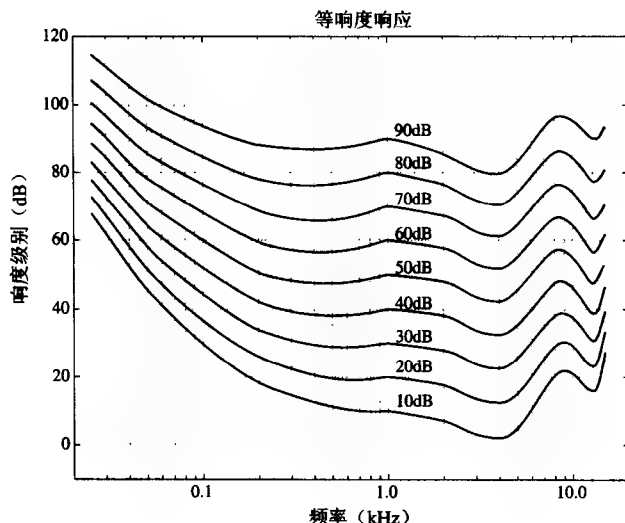


图 14-1 Fletcher-Munson 的人耳等响度响应曲线 (Robinson 和 Dadson 重新测量)

所有曲线是这样排列的，使接收到的声音响度值能给出与纯音色的 1kHz 的响度等级相同的响度。因此，1kHz 这一点的响度值和总是和纵轴的 dB 值相同的。例如，最下面的一条曲线就是 10 方。这条曲线上的所有音色的响度和 10dB，1000Hz 的音色的响度相同。Robinson 和 Dadson[1] 绘制的图比 Fletcher 和 Munson[2] 原来的图更加准确。

等响度曲线的主要思想是音色是以某个频率和可度量的响度级产生，然后人类以一定比率接收响度。如最下部的曲线所示，所有在 20Hz 到 15kHz 之间的纯音色都以在纵轴给出的一个音量值产生，这个音色将以 10dB 的响度级感知[1]。第二条曲线显示了对于每一个以 20dB 感知的纯音色的幅值，以此类推。最上面的一条曲线是以 90dB 被感知的。

举例来说，在 5 000Hz 处，当声源实际上只有 5dB 时，我们感知到的响度具有 10 方。注意，在 4kHz 处，当激励实际上只有大约 2dB 的时候，我们感受到声音有 10dB。为在 10kHz 处感受到 10dB 的效果，我们不得不产生一个 20dB 的激励。人耳朵在 2kHz 到 5kHz 之间的敏感度比 6kHz 以上更加敏感。

在最低频率上，如果声源强度为 10dB，1kHz 的音色也会在 10dB 发声；但是，若频率更低，那么 100Hz 的音色必须在 30dB 处才能发声，比在 1kHz 处高 20dB！所以我们对于较低的频率不敏感。产生这种现象的原因是我们的耳道对 2.5~4kHz 的频率有放大作用。

注意，当整体的响度增加时，曲线会变得平坦。如果声音足够响的话，我们对于几百赫兹的低频声音的敏感程度是基本上相同的，而且我们对于低频声音的感知程度要好于高音量的高频。

因此，在舞厅中，声音大的音乐听起来比安静的音乐更清楚，因为我们实际上只是听到了低频的声音，而没有听到高频的声音。（“响度”在某些声音系统中的切换只是提升低频和高频。）然而，当声音超过 90dB 时，人们会感觉到不舒服。通常，城市中地铁运行时的声音大约是 100dB。

### 14.1.2 频率遮掩

一种音色是如何干扰另外一种音色的呢？一种频率在什么程度上能盖过另一种频率？这个问题我们可以通过遮掩曲线获得答案。而且，遮掩回答了在能够听到真正的音乐之前我们能够忍受何种程度的噪声的问题。有损的音频数据压缩算法（比如 MPEG 声音或者在电影制作中常用的杜比数字（AC-3）编码）都去掉了某些被遮掩的声音，因此减少了总的信息量。

关于遮掩，有以下几种情况比较常见：

- 一个较低的音色可以有效的遮掩（使它无法被听到）另一个较高的音色。
- 反之则不然。一个较高的音色不一定能很好地遮掩一个较低的音色。音色实际上可以遮掩较低频率的声音，但是并不像遮掩较高频率的音色那样有效。
- 遮掩的音色越强，它的影响就越广，也就是它能够遮掩的频段就越宽。
- 因此，如果两种音色在频率上相差较大，就几乎不会有遮掩发生。

#### 1. 听觉阈值

图 14-2 显示了人类对于纯音的听觉阈值。为获得这样一个图，需要产生一种特定频率的声音，比如 1kHz。在一个安静的房间里或者使用耳机先将音量降低到零，然后逐渐升高直到声音刚刚能够被听到。用这种方法可以为所有能够听到的频率生成对应的数据点。

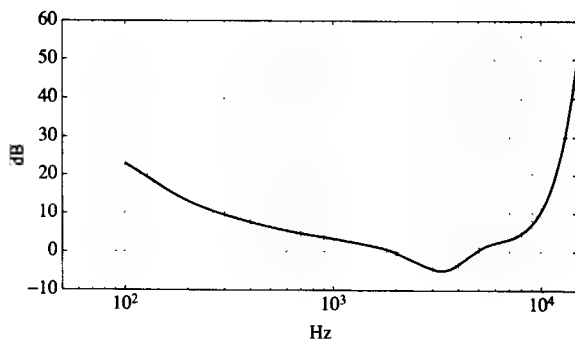


图 14-2 人类对于纯音的听觉阈值

398

听觉阈值曲线上的点表明，如果一种声音超过图中所示的 dB 大小，比如超出 2dB 的 6kHz 音调，就能够被听到。否则，我们不能听到它。调高 6kHz 音调音量，使它等于或者超过曲线意味着我们能够区分这种声音。

关于这条曲线的近似公式，如下[3]所示：

$$\text{Threshold}(f) = 3.64(f/1000)^{-0.8} - 6.5e^{-0.6(f/1000-3.3)^2} + 10^{-3}(f/1000)^4 \quad (14.1)$$

阈值单位是 dB。因为单位 dB 是一个比值，所以我们需要确定以何种频率作为原点 (0,0)。在式 14.1 中，这个频率是 2000Hz，即当  $f=2\text{kHz}$  时， $\text{Threshold}(f)=0$ 。

#### 2. 频率遮掩曲线

我们通过产生一种高音量的纯音，比如 1kHz 的频率，并确定这种音色如何影响我们听到临近频率音色的方法，来研究频率遮掩。为了完成这项工作，我们将产生一个 1kHz 的遮掩音色，并把它固定在 60dB，然后提高临近音色值，比如 1.1kHz，直到它能够被听到。图 14-3 中的阈值显示了这个能够被听到的值。

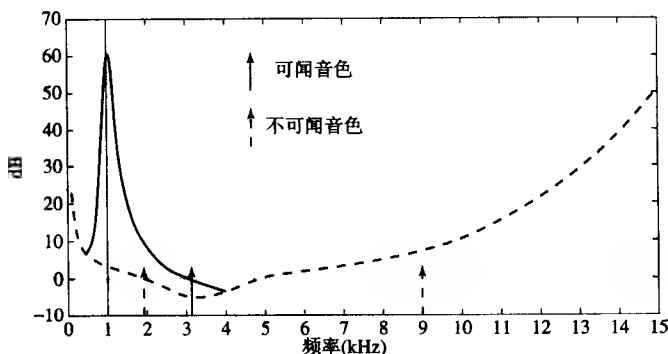


图 14-3 1kHz 遮掩音色对于人类听觉阈值的影响

注意，这条遮掩曲线图只用于一种遮掩音色，这一点很重要。如果使用其他遮掩音色，图形就会改变。图 14-4 说明了这一点，遮掩音色的频率越高，它所影响的范围越广。

举例来说，假设我们在存在一个 4kHz 遮掩音色的时候产生一个 6kHz 的音色，那么遮掩音色会大幅度提升阈值曲线。因此，在 6kHz 的临近频率上，我们现在必须增加 30dB 才能把它和 6kHz 的音色区分开。

实际上，如果一个信号能够被分解为频率，那么它的频率将会被部分遮掩，只有能够听到的部分将会被用来设置量化噪声阈值。

399

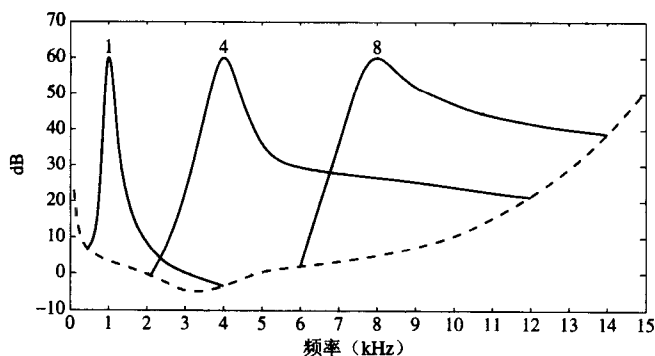


图 14-4 遮掩音色在 3 种不同频率上的效果

### 3. 临界频段

由于当有其他声音时，人类听觉系统具有在大约一个临界频段内能更好地分解声音的特性，因此人类听力范围自然地分为多个临界频段。听觉具有一个有限的、取决于频率的分解。由[4]可知，“在一个复杂的音色里，临界带宽与两部分的最小频率差异相对应，这个差异使得它们仍然可以被独立地听到……临界带宽表示人耳分辨同时出现的音色或者部分的能力。”

在低频端，临界频段小于 100Hz，但在高频时，频带会大于 4kHz。这确实是另外一种感知不一致的情况。

实验表明，当遮掩频率低于 500Hz 时，临界频段的带宽基本保持恒定，其宽度大约为 100Hz。然而，当频率大于 500Hz 时，临界频段的宽度随着频率的提高呈近似线性的增长。

一般来说，听力的音频频率范围可以被分为 24 个临界频段（编码应用通常使用 25 个临界频段），如表 14-1 所示。

虽然一般对于临界频段是如此定义的，但是我们的听觉器官实际上是在某种程度上调谐到一

个确定的临界频段。因为听觉取决于内耳的物理结构，所以对于这种结构使之共鸣效果最好的频率是很重要的。频率遮掩是耳朵结构在遮掩频率和相邻频率处变得“饱和”的结果。

因此，耳朵的工作原理和一组带通滤波器的工作原理相似，每个带通滤波器只允许限定范围的频率通过而阻截其他波段的频率。证明上述观点的实验是基于恒定音量声音的一种现象，这种现象是如果它跨越了两个临界频段的分界，那么将会比它完成包含在一个临界频段内听起来更响。因此，由于遮掩的作用，人耳在一个临界频段内不能够很好的区分声音。

400

表 14-1 临界频段及其带宽

频带号	能量范围	中心 (Hz)	上限 (Hz)	带宽 (Hz)
1	—	50	100	—
2	100	150	200	100
3	200	250	300	100
4	300	350	400	100
5	400	450	510	110
6	510	570	630	120
7	630	700	770	140
8	770	840	920	150
9	920	1 000	1 080	160
10	1 080	1 170	1 270	190
11	1 270	1 370	1 480	210
12	1 480	1 600	1 720	240
13	1 720	1 850	2 000	280
14	2 000	2 150	2 320	320
15	2 320	2 500	2 700	380
16	2 700	2 900	3 150	450
17	3 150	3 400	3 700	550
18	3 700	4 000	4 400	700
19	4 400	4 800	5 300	900
20	5 300	5 800	6 400	1 100
21	6 400	7 000	7 700	1 300
22	7 700	8 500	9 500	1 800
23	9 500	10 500	12 000	2 500
24	12 000	13 500	15 500	3 500
25	15 500	18 775	22 050	6 550

#### 4. Bark 单位

因为受遮掩影响的频率范围随着频率的增高而增宽，所以定义一种新的频率单位是很有用的，这样在使用这种新单位表示时，每一条遮掩曲线（上面图 14-4 的一部分，在安静阈值之上）拥有相同的带宽。

401

定义的新的单位称为 Bark，是以一位声音科学家 Heinrich Barkhausen (1881—1956) 而命名的。一个 Bark 单位对应于任何遮掩频率 $[6,7]$ 的一个临界频段的宽度。图 14-5 显示了临界频段，它的频率（横坐标）是使用 Bark 单位表示的。

频率 $f$ 和它对应的使用 Bark 单位表示的临界频段数值 $b$ 之间的转换关系如下：

$$\text{临界频段值 (Bark)} = \begin{cases} f/100 & f < 500 \\ 9 + 4 \log_2(f/1000) & f \geq 500 \end{cases} \quad (14.2)$$

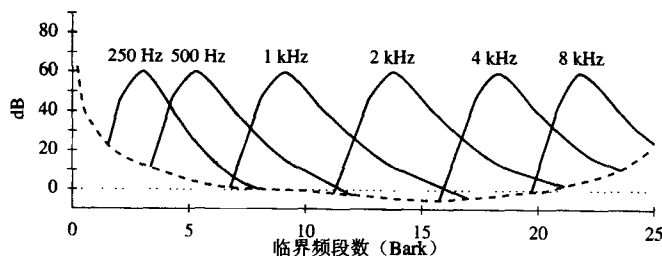


图 14-5 遮掩音色的效果, 使用 Bark 单位表示

使用这种新的频率单位度量, 当  $f = 500\text{Hz}$  时, 临界频段数值  $b$  等于 5。将该频率倍频, 即遮掩频率是  $1\text{kHz}$  时, Bark 值提高到 9。另一个度量 Bark 的公式如下:

$$b = 13.0 \arctan(0.76f) + 3.5 \arctan(f^2/56.25) \quad (14.3)$$

其中,  $f$  的单位为 kHz,  $b$  的单位为 Bark。将公式倒推即可求出对应于特定的 Bark 值  $b$  的频率 (以 kHz 为单位):

$$f = [(\exp(0.219 \times b)/352) + 0.1] \times b - 0.032 \times \exp[-0.15 \times (b-5)^2] \quad (14.4)$$

形成两个临界频段之间边界的频率, 是由整数 Bark 值给出的。对于给定的中心频率  $f$ , 临界带宽 ( $df$ ) 也可以用下式估计[8]:

$$df = 25 + 75 \times [1 + 1.4(f^2)]^{0.69} \quad (14.5)$$

其中  $f$  的单位为 kHz,  $df$  的单位为 Hz。

设置 Bark 单位的目的是定义一种更均匀一致的频率单位, 使得每个临界频段的宽度使用 Bark 表示时都大致相等。

### 14.1.3 时间遮掩

回想一下, 在跳舞之后我们需要很长一段时间听力才能够恢复正常。一般来说, 任何大的音色都会导致内耳内的听觉接收器 (很小的像头发一样的结构被称作 cilia) 变饱和, 它们需要一段时间才能恢复 (很多感知系统也会有这样的时间延缓方式——比如, 眼睛中的接收器拥有类似的电容效应)。

为了量化这种行为, 我们可以使用另外一种遮掩实验来度量听觉的时间敏感性。假设我们再生成一个遮掩音色和一个临近的音色, 遮掩音色的频率为  $1\text{kHz}$ , 音量为  $60\text{dB}$ , 临近音色是  $1.1\text{kHz}$ , 音量为  $40\text{dB}$ 。因为临近的测试音色被遮掩, 所以不能够听到它。但是当遮掩音色被去掉时, 我们可以再次听到  $1.1\text{kHz}$  的音色, 只不过在一段时间之后才能听到。实验在关掉遮掩音色后, 比如 10 毫秒之后, 关掉测试音色的方法进行。

延时时间的长度应调整到能够区分测试音色所需的最小的时间长度。一般来说, 测试音色越响, 我们的听觉克服遮掩而听到遮掩音色的时间就越短。图 14-6 显示了这种效果: 在  $60\text{dB}$  遮掩音色产生后, 我们需要 500 毫秒的时间辨别出安静的测试音色。当然, 这条曲线会随着遮掩音色频率的不同而发生变化。



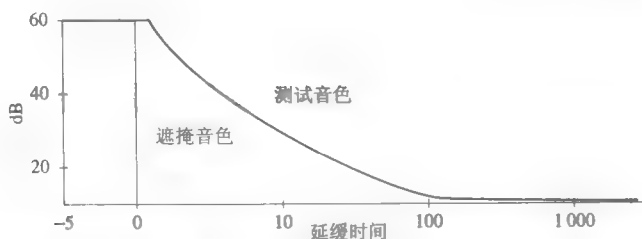


图 14-6 测试音色越响，当遮掩音色被去除时听到测试音调所需要的时间越短

测试音色的频率越接近遮掩音色的频率，它被遮掩的程度越高。因此，给定一个遮掩音色，我们能够看到二维时间遮掩情况，如图 14-7 所示。测试音色的频率越接近遮掩音色的频率，当遮掩音色停止后所需要的时间就越接近，测试音色无法被听到的可能性就越大。图中显示了频率和时间遮掩的整体效果。

饱和现象也和使用多长时间的遮掩音色有关。图 14-8 显示，如果一种遮掩音色保持时间（200 毫秒）比另外一种的时间（100 毫秒）更长，那么听到测试音色需要的时间就更长。

正如它可以遮掩其他刚好在它产生之后产生的信号（后遮掩）一样，某个信号甚至可以遮掩刚好在它产生之前的信号（前遮掩）。

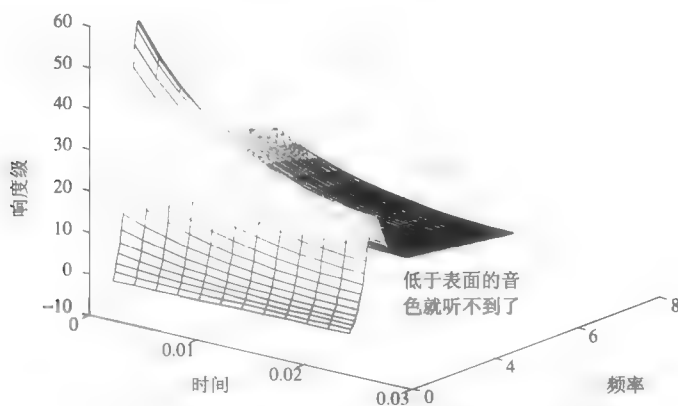


图 14-7 时间遮掩的效果取决于时间和频率临近的程度

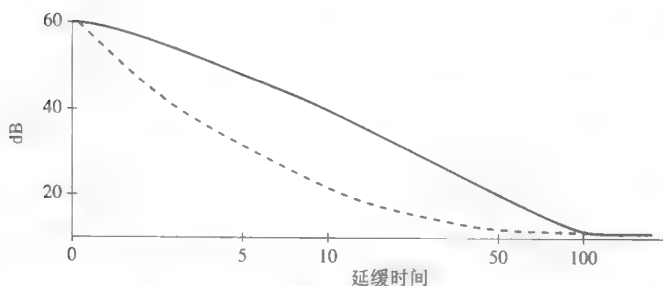


图 14-8 时间遮掩的效果也和遮掩音色所应用的时间有关。实线表示遮掩音色保持 200 毫秒；虚线表示遮掩音色保持 100 毫秒

前遮掩的有效时间间隔更短(2~5 毫秒),它在这方面比后遮掩(一般 50~200 毫秒)更有效。

MPEG 音频压缩利用前面提到的因素构成一个庞大的多维查找表。它使用这个表来传送被遮掩的频率成分,这些成分或因频率遮掩,或者因时间遮掩,或者被两者同时遮掩,从而获得更少的位数。

## 14.2 MPEG 音频

MPEG 音频先对输入应用一个滤波器组,把输入分解得到其频率成分。同时,对数据应用一个心理声学模型,这个模型在位分配块中使用。然后,被分配的位数用于量化从滤波器组获得的信息。最后的结果是量化提供压缩,位被分配到最需要它们的地方以便将量化噪声降低到可听到的级别之下。

### 14.2.1 MPEG 的层

MP3 是一种常用的音频压缩标准。数字“3”代表三层,“MP”代表 MPEG-1 标准。让我们回忆一下第 11 章中的 MPEG 视频压缩算法。不过,MPEG 标准实际上刻画多媒体的三个不同方面:音频、视频和系统。MP3 是 MPEG 第一阶段的音频部分。它在 1992 年发布,并且为 1993 年发布的国际标准 ISO/IEC11172-3 奠定了基础。

MPEG 音频使用 3 个向下兼容的音频压缩层,每一层都能够理解比它低的层。每一层提供更复杂的心理声学模型应用,相应地,为给定级别的音频质量提供更好的压缩。但是,随着复杂性的增加,压缩率的提高,会带来更多的延时。

MPEG 音频中的第一层到第三层是兼容的,因为所有的层包含相同的文件头信息。

第一层的质量非常好,可以提供相当高的码率。数字音频磁带一般就使用第一层。第二层更复杂一些,并且通常用于数字音频广播。第三层(MP3)最复杂,它的最初的目标是在 ISDN 线路上进行音频的传输。每一层都使用不同的频率变换。

大部分复杂性不是在解码端而是在编码端增加的,这也是 MP3 播放器如此流行的原因。第一层结合了最简单的心理声学模型,而第三层使用了最复杂的心理声学模型。其目标是权衡好质量和码率之间的关系。“质量”是用收听测试得分来定义的(心理学家在这方面拥有统治地位),其中质量度量的定义如下:

- 5.0=“清晰的”——觉察不到与原始的信号有差异;相当于使用 14 或者 16 位 PCM 编码的 CD 质量的音频。
- 4.0=能够感觉到差异,但是不会让人感到烦扰。
- 3.0=稍微让人感觉烦扰。
- 2.0=感觉烦扰。
- 1.0=感觉非常烦扰。

(现在就非常科学了!)在每个通道 64kbps 的传输率下,第二层的分数为 2.1~2.6,而第三层得分为 3.6~3.8。所以第三层取得了很大的进步,但是仍然不够完美。

### 14.2.2 MPEG 音频策略

压缩是肯定需要的,因为即使是音频也会占用相当大的带宽:CD 音频的采样频率是 44.1kHz,每通道 16 位,所以两个通道需要大约 1.4Mbps 的码率。MPEG-1 的目标是大约 1.5Mbps,其中 1.2Mbps 用于视频,而 256kbps 用于音频。

MPEG 压缩方法和量化有关,当然,我们也能看出人类听觉系统在临界频段宽度内,在接收到的响度和测试频率的能听度这两个方面,并不是准确的。编码器使用了一组滤波器,通过计算信号值窗口的频率变换,首先分析音频信号的频率(频谱)成分。滤波器组把信号分解成很多子

带。第一层和第二层编/解码使用积分映射滤波的方式，而第三层编/解码增加了一个 DCT（离散余弦变换）。对于心理声学模型，可以使用傅里叶变换。

然后就可以使用频率遮掩了，使用心理声学模型来估计可以被察觉的噪声级别的临界值。在它的量化和编码阶段，通过去掉不能听到的频率和依照在遮掩之后剩余声音的级别伸缩量化步长，编码器可平衡遮掩行为和可用的位数。

复杂的模型可以考虑不同频率中心的临界频段的实际宽度。在临界频段内，我们的听觉系统不能很好地分辨临近的频率，而通常会把它们混和起来。正如前面提到的，能够听到的频率通常被分为 25 个主临界频段，这是听觉临界频段启发的结果。

然而，为保持设计的简单性，模型通过 32 个重叠子带对所有频率分析滤波器使用了一种统一宽度[9,10]。这意味着，在低频处，每一个频率分析“子带”覆盖了听觉系统的临界频段的宽度，然而对于高频来说并不是这样，因为临界频段的宽度在低端小于 100Hz，而在高端要大于 4kHz。对于每一个频段，在遮掩级别之上的声级表明必须分配多少位来编码信号值，从而使得量化噪声能够被控制在遮掩级别之下而不会被听到。

在第一层中，心理声学模型只使用了频率遮掩。码率范围从 32kbps（单声道）~448kbps（立体声）。用 256~384kbps 的码率可以得到类似 CD 的立体声音质。第二层使用一些时间遮掩方法聚集了更多的样例，并且分析在当前样例块和其之前或者之后的样例块之间的时间遮掩。其码率可以达到 32~192kbps（单声道）和 64~384kbps（立体声）。立体声 CD 音频质量需要大约 192~256kbps 的码率。

406

但是，时间遮掩对于压缩来说并没有频率遮掩重要，这也是为什么有时完全不使用低复杂性编码器的原因。第三层关注低码率应用，使用具有不均匀子带宽度的更复杂的子带分析。它也添加了不均匀量化和熵编码。码率被标准化为 32~320kbps。

### 14.2.3 MPEG 音频压缩算法

#### 1. 基本算法

图 14-9 显示的是一个基本的 MPEG 音频压缩算法。它通过滤波器组，把输入分解成 32 个子带。这是一个线性的过程，就好像输入是一组 32 个 PCM 样本，按时间顺序采样，而产生的输入是 32 个频率的系数。如果采样频率是  $f_s$ ，令  $f_s = 48\text{kps}$ （每秒千样本数，即 48kHz），那么由奈奎斯特定理可知，最高的映射频率将会是  $f_s/2$ ，因此映射带宽被分为 32 个等宽的段，每段的宽度是  $f_s/64$ （这些段有些相互重叠）。

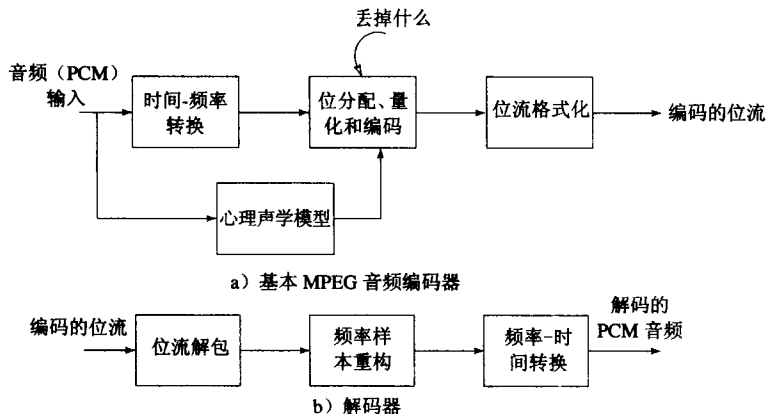


图 14-9

407

在第一层编码器中，32 PCM 值的集合被首先组合成 12 组的集合，每组 32 个。因此，编码器具有固有的时间延时，等于聚集 384（即  $12 \times 32$ ）个样本的时间。例如，如果采样是以 32kbps 进行的，那么 12 毫秒的持续时间是必要的，因为每毫秒传送一个含有 32 个样本的 1 个集合。这 12 个样本的集合，每一个大小是 32，被称作段。组合的关键在于，在频率分析完成后，同时分析在 32 个子带中的 12 个值的集合，然后分析所有 12 个数值总体的基础量化值。

延时实际上比所需要的聚集 384 个样本的时间还要长一点，因为还需要头信息。而且，还可以有辅助数据，比如多语数据和环绕声音数据。较高的层还可以分析比 384 更多的样本，因此子带样本（SBS）的格式也可以加入，SBS 使用一种合成的数据帧，如图 14-10 所示。头信息中包含同步编码（12 个 1——111111111111）、使用的采样频率、码率和立体声信息。帧格式也包含所谓的“辅助”（额外）信息空间（事实上，一个 MPEG-1 音频解码器可以至少部分解码一个 MPEG-2 音频位流，因为文件头以 MPEG-1 头开始，然后将 MPEG-2 数据流放在 MPEG-1 存放辅助数据的地方）。

头部	SBS 格式	SBS	辅助数据
----	--------	-----	------

图 14-10 MPEG 音频帧示例

MPEG 音频可以用于处理立体声，当然也可以处理单声道。一个特殊的连接立体声模式通过考虑两个立体声通道之间的冗余产生一个流。这是一个视频信号合成的音频版本。它能够处理二重单频道——两个通道独立编码。这在并行处理音频的时候十分有用——比如，处理两个语音流（一个是英语，另外一个西班牙语）。

把  $32 \times 12$  段视为一个  $32 \times 12$  的矩阵。算法下一阶段要考虑比例，以便设置合适的量化级别。对于 32 个子带的每一个子带，找到数组中各行对应的 12 个样本的最大幅度，它就是那个子带的缩放因子。随后，将这个最大值传递给算法的位分配块，一起传递的还有 SBS（子带样本）。位分配块的重点是确定如何为量化子带信号分配可用的编码位，从而使可听到的量化噪声降低到最小程度。

我们都知道，心理声学模型是相当复杂的，比起一个查找表集合要复杂得多（实际上这种模型并没有在规范中加以标准化——它的形成的是音频编码器的“艺术”内容，并且是所有的编码器不都一样的主要原因）。在第一层中，包含一个确定的阶跃，以基本上确定每种频段更像一个音色还是噪声。根据这个决定和缩放因子，我们计算每个频段的遮掩阈值，并且与听觉阈值进行对比。

模型的输出由一组称作信号遮掩比（Signal-to-Mask Ratio, SMR）的值所组成，它们标记了幅度低于遮掩级别的频率成分。SMR 是在每一个频段中短期信号的能量与最小子带遮掩阈值的比值。SMR 给出了所需的幅度分解，因此也可以控制将要分配给子带使用的位。在确定 SMR 之后，可使用前面讨论的缩放因子来设置量化级别，使得量化误差自身降到遮掩级别之下。这样可以确保将更多的位用于听觉最敏感的区域。总之，当使量化噪声不会被听到所需的位越少，编码器在临界频段将使用的位数也越少。

408

缩放因子首先使用 6 位进行量化。然后量化每个子带的 12 个值。使用 4 位，在使用一个迭代的位分配方案之后，对每一个被传输的子带进行位分配。然后数据使用适合每个子带的位深度加以传输。总而言之，数据是由量化缩放因子和组成一个称为子带样本格式的 12 个码字组成的。

在解码器端，对值进行逆量化，重新建立 32 个样本幅度。它们要通过一个合成滤波器组，合成滤波器的作用就是重建 32 个 PCM 样本的集合。注意，在解码器端并不需要心理声学模型。

图 14-11 显示了如何组织样本。一个第二层或者第三层的帧实际上对于每一个子带聚集的样本多于 12 个，即一帧包含 1152 个样本，而不是 384 个。

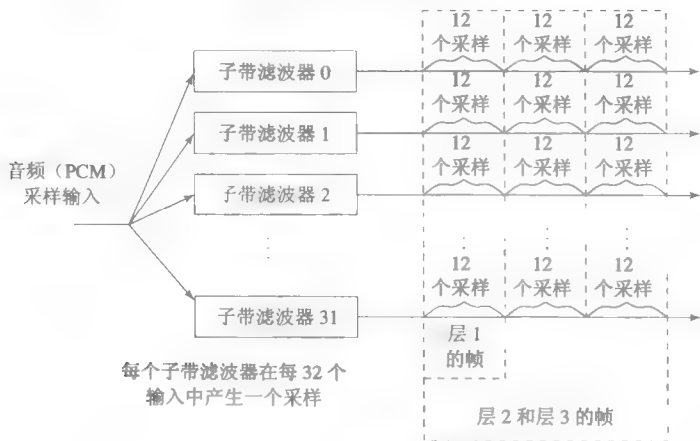


图 14-11 MPEG 音频帧的大小

2. 位的分配

位的分配算法并不是标准的一部分，因此可以使用很多种可能的方法来完成它。它的目标是保证所有的量化噪声低于遮掩阈值。然而，对于低码率的情况通常就不是这样了。心理声学模型在这些情况下发挥了作用，从可用的位数中分配更多的位给最能有效提高分辨率的子带。一个常见的方案如下所示。

对于每一个子带，心理声学模型计算信号遮掩比（以 dB 为单位）。MPEG 音频标准的查找表也提供了一个 SNR（信噪比）的估计，它假设对一个给定量化器等级进行量化。

409

然后，遮掩噪声比（Mask-to-Noise Ratio, MNR）的定义如下：

$$\text{MNR}_{\text{dB}} = \text{SNR}_{\text{dB}} - \text{SMR}_{\text{dB}} \quad (14.6)$$

如图 14-12 所示。所有子带上的最低的 MNR 已经确定，并且增加了分配给此子带编码的位数。而后得出一个新的 SNR 估计，整个过程继续迭代，直到没有剩余位能够用来分配为止。

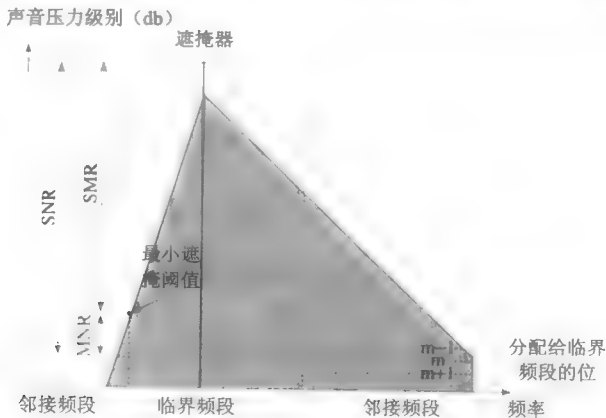


图 14-12 遮掩噪声比和信号遮掩比。一个定性的 SNR、SMR 和 MNR 的图像，使用一个主要的遮掩和给特定的临界频段分配  $m$  位的方法

遮掩计算与子带滤波并行执行,如图 14-13 所示。计算遮掩曲线需要输入准确的信号分解,使用离散傅里叶变换(DFT)。频谱通常使用 1024 点的快速傅里叶(FFT)变换来计算。

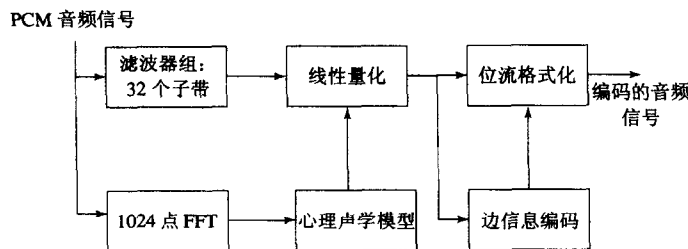


图 14-13 MPEG-1 音频的第一层和第二层

在第一层中,16 个统一量化器被提前计算好,对于每个子带,量化器给出了被选择的最低失真。量化器的索引以每个子带边信息的 4 位被发送出去。每一个量化器的最大分辨率是 15 位。

### 3. 第二层

MPEG-1 音频编/解码的第二层包含对减少码率和提升质量的小改进,其代价是增加了复杂性。它不同于第二层主要是对 12 个样本的三组在每帧内进行编码,并且利用时间遮掩和频率遮掩。它的一个优势就是如果三个组的缩放因子比较相似,那么其中一个缩放因子可以用于所有的三个组。但是,在滤波器组中使用三个帧(之前、现在、之后),每个通道共有 1152 个样本,只是近似考虑了时间遮掩。

而且,如果使用更长的时间窗,那么在为缓慢变化的声音建模方面心理声学模型更加适用。我们使用 36 个样本长度的窗而不是 12 个进行位分配,量化器的分辨率也从 15 位上升到 16 位。为了保证精度提高不会使压缩效果变差,对于更高的子带,我们选择较少的量化器的数量。

### 4. 第三层

第三层(即 MP3)使用类似于第一层和第二层的码率,但是它提供更好的音频质量,其代价是增加复杂性。

第三层也使用类似于第二层的滤波器组,只是现在更加与感知的临界频段紧密相连了,这种粘着是通过使用一组不相等频率的滤波器实现的。这一层也考虑了立体声冗余的问题。它还使用了一个改进的傅里叶变换,即用修订离散余弦变换(MDCT)来处理离散余弦变换(DCT)在窗口边界所存在的问题。离散傅里叶变换可以阻止边界效应的发生。当这样的数据被量化,然后被逆变换回时间域的时候,起点和终点的样本块可能会和先前的或者随后的块不协调,从而发生听到周期性噪声的问题。

在式(14.7)中显示的 MDCT 通过引入 50% 的重叠帧的方法去除了这种效果。

$$F(u) = 2 \sum_{i=0}^{N-1} f(i) \cos \left[ \frac{2\pi}{N} \left( i + \frac{N/2+1}{2} \right) (u+1/2) \right], u=0, \dots, N/2-1 \quad (14.7)$$

MDCT 也为遮掩和频率分配操作给出了更好的频率分辨率。另外,窗口大小可以从 36 个样本降为 12 个样本。即便如此,因为窗口有 50% 重叠,一个 12 个样本的窗口仍然可以包含 6 个额外的样本。一个大小为 36 个样本的窗口包含 18 个额外的样本。因为低频时,一个频率更像一个音色而不是噪声,不需要对它们仔细分析,所以我们可以使用一种混和的模式,对频率最低的两个子带使用 36 点的窗口而对其他子带使用 12 点的窗口。

而且,MDCT 系数依照听觉系统的实际临界频段进行分组,而且缩放因子频段就是根据这些

系数来计算的，而不是把子带的统一宽度赋给缩放因子。

通过使用熵编码和利用不均匀量化器的方式保存更多的位数。并且，最终使用一种不同的位数分配方案，它分为两部分。首先，使用一个嵌套循环，内层循环负责调节量化器的形状，而外层循环则评价位数配置的失真度。如果错误率（“失真”）太高，则增大缩放因子频段。其次，位数池从不需要位的帧中积蓄位，然后把它们分配到需要位的帧中。图 14-14 显示了 MPEG 音频第三层编码的框图。

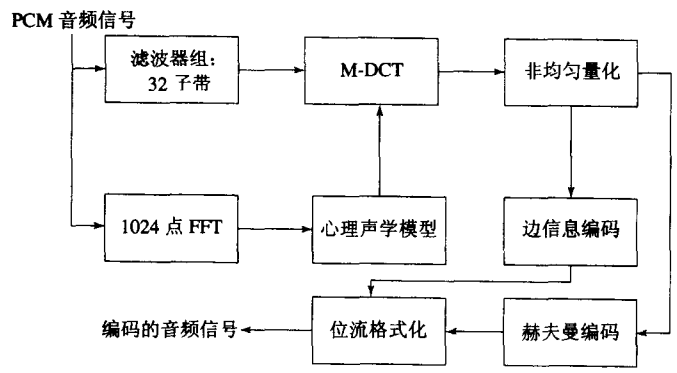


图 14-14 MPEG-1 音频的第三层

表 14-2 显示出多种已经实现的 MP3 压缩率。特别要注意的是，CD 音质已经实现 12:1~8:1 的压缩率（即码率为 128~192kbps）。

表 14-2 MP3 压缩性能

声音质量	带 宽	模 式	压缩率
电话	3.0 kHz	单声道	96:1
好于短波	4.5 kHz	单声道	48:1
好于 AM 广播	7.5 kHz	单声道	24:1
类似 FM 广播	11 kHz	立体声	26:1~24:1
接近 CD	15 kHz	立体声	16:1
CD	>15 kHz	立体声	14:1~12:1

14.2.4 MPEG-2 高级音频编码

MPEG-2 标准被广泛使用，因为它是 DVD 的标准传播媒介，而且它还拥有音频成分。MPEG-2 高级音频编码（AAC）标准[11]的目标是剧场的清晰声音重现。它可以以 5 通道 320kbps 进行传播，因此声音可以从 5 个方向而来：左、右、中心、左环绕和右环绕。所谓的 5.1 通道系统还包含低频增强（LFE）通道（“低音炮”）。另一方面，MPEG-2 AAC 还可以以低于 128kbps 的码率传播高音质立体声。它是一种用于可录 DVD-音频（DVD-AR）格式的音频编码技术，已经在两个北美卫星广播服务之一的 XM Radio 上使用。

MPEG-2 音频可以支持最多 48 个通道，采样率为 8~96kHz，每通道码率最高可达 576kbps。和 MPEG-1 一样，MPEG-2 支持三种不同的“设置”，但是目的不同。它们是主要（Main）、低复杂度（LC）和可伸缩采样率（SSR）。LC 设置需要的计算少于 Main 设置，但是 SSR 设置会分解信号使得不同解码器可以使用不同码率和采样率。

这 3 种设置所遵循的方案几乎相同，只有少许修改。首先，在有 2048 个样本的“长”窗口

或者 256 个样本的“短”窗口进行 MDCT 变换。随后使用时域噪声整形 (TNS) 工具对 MDCT 系数进行滤波, 目标是降低前遮掩效果并获得具有稳定音调的更好的编码信号。

MDCT 系数随后被分成 49 个缩放因子频段组, 这些组大致和人类听觉系统的临界频段的良好分辨率版本一致。在进行频率变换的同时, 使用 MPEG-1 中类似的心理声学模型, 求出遮掩阈值。

Main 设置使用一个预测器。使用基于之前的两帧, 而且最高只影响 16kHz 频率系数的方法, MPEG-2 从频率系数中减去预测值, 这个步骤确实可以降低失真度。量化器遵循两条规则: 将失真度保持在遮掩阈值之下, 并且使用位池, 保持每帧使用的平均位数在控制范围之内。量化器使用缩放因子 (它可以用于增大某些缩放因子频段) 和不均匀量化。MPEG-2 AAC 也对缩放因子和频率系数使用熵编码。

同样地, 位分配也使用嵌套循环。内层循环采用非线性量化器, 然后对量化数据应用熵编码。如果达到当前帧的位界限, 那么增加量化器的步长大小以使用更少的位。外层循环决定对于每一个缩放因子频段来说, 失真度是否在遮掩阈值之下。如果频段失真过于严重, 那么就提高此频段的 SNR, 当然代价就是使用更多的位。

在 SSR 设置中, 使用一组多相积分滤波器 (PQF)。这个短语的含义是信号首先被分成四个宽度相等的频段, 然后使用 MDCT。第一步的重点是, 如果必须降低码率的话, 解码器可以决定忽略四个频率中的哪一部分。

#### 14.2.5 MPEG-4 音频

MPEG-4 音频把多种不同音频成分整合成了一种标准: 语音压缩、基于感知的编码器、文语转换和 MIDI。主要的一般音频编码器 MPEG-4 AAC[12]和 MPEG-2AAC 标准很相似, 只有一些小的变化。

##### 1. 感知编码器

MPEG-4 的一个改进是引入感知噪声置换模块, 它考虑在 4kHz 以上的比例因子频段, 并且包含关于声音是噪声还是音色的判定。一个类噪声缩放因子频段本身不能被传输, 而是只传送它的能量, 而且频率系数被设置为零。然后解码器插入具有此能量的噪声。

另一个改进是包含位分片算术编码 (BSAC) 模块。这是一种用于提高码率可伸缩性的算法, 它是通过允许解码器端只使用 16kbps 基准输出 (自最小值端 1kbps 的步长) 从而对 64kbps 流解码来实现的。

MPEG-4 音频包含的第二个感知音频编码器被称作变换域加权插入矢量量化 (TwinVQ) 的向量量化方法。它针对低码率, 允许解码器丢弃位流的一部分来实现码率和采样率的可调节性。MPEG-4 音频的基本策略是解码器使用的音频工具的数量要和带宽的要求一致。

##### 2. 结构化编码器

为了实现低码率传输, MPEG-4 使用了一种称为合成/自然混合编码 (SNHC) 的方法。它的目标是把“自然的”多媒体序列、音频和视频及其相关内容综合起来。在音频中, 后者叫作“结构化”音频。这个想法主要用于低码率操作, 我们可以只传递一个指向我们正在使用的指针然后传递音频模型的参数。

在视频上, 基于模型这样的方法可能涉及传递脸部动画数据而不是自然的脸的视频帧。在音频上, 我们可以传递英语已经被建模的信息, 然后在传递英语的基本音 (音素) 的同时, 传递其他规定了持续时间和音调的类似装配的编码。

MPEG-4 使用工具箱方法, 允许相当多的模型规范。比如, 文语转换 (Text-to-Speech, TTS) 是一种极低码率的方法, 而且我们不需要知道说话者实际上如何发音就可以使用。假设我们可以



从相当低码率的信息中派生出脸部动画参数，我们可以实现一个相当低的码率视频会议系统。

414

结构化音频中的另一个“工具”称为结构化音频命令语言（SAOL，发音为“sail”），它可以应用简单的音频分析规范，包括混响这样的特殊效果。

总之，结构化音频利用了音乐中的冗余，大大压缩了音频描述。

### 14.3 其他商业音频编解码器

表 14-3 总结了其他现代常用音频编码器的目标的码率范围和主要特性。它们和 MPEG-2 编码有很多相似之处。

表 14-3 音频编码系统的对比

编解码器	位率 (kbps/声道)	复杂度	主要应用
DolbyAC-2	128-192	低 (编码器/解码器)	点到点, 有线
DolbyAC-3	32-640	低 (解码器)	HDTV, 有线, DVD
SonyATRAC	140	低 (编码器/解码器)	小磁盘

### 14.4 未来: MPEG-7 和 MPEG-21

回想一下，MPEG-4 的目标是利用对象来压缩。MPEG-4 音频拥有很多有趣的特性，比如 3D 的声音定位、MIDI 的整合、文语转换、不同码率的不同编解码器以及使用复杂的 MPEG-2 AAC 编解码器。但是，新的 MPEG 标准将主要着眼于“查找”：假如多媒体确实要根据对象进行编码，我们如何能够找到对象？

MPEG-21[13]的公式是一个正在进行的努力，其目标是努力制定一套从消费者视角出发的多媒体框架标准，特别强调互操作性。但是，我们可以讨论一些关于 MPEG-7 方法如何描述结构化的音频模型的细节，以达到提升音频搜索方便性的效果。

MPEG-7 提供了一种针对视听多媒体序列进行元数据标准化的方法，其正式名称是多媒体内容描述接口。MPEG-7 用于表示关于多媒体信息的信息。

在音频方面，目标是方便音频内容的表现和通过音色或其他描述符进行查找。因此，研究人员正在致力于开发一种能够有效描述并且帮助在文件中查找特定音频的描述符。这些都需要人类分析或者自动内容分析，并且不是仅仅以低层结构（如旋律）为目标，而是从相关的结构或者语义内容中提取信息。

一个 MPEG-7 支持的示例应用是自动语音识别（ASR）。语言理解也是 MPEG-7 “内容”的一个目标。从理论上说，MPEG-7 将允许按照口语或者视觉事件上进行搜索：“为我找到哈姆雷特在第几章说过‘生存还是毁灭’。”但是，MPEG-7 的描绘一个完整的、结构化的音频模型的目标是没有办法完成的。

415

不过，低层特性是重要的。关于这些工作的最新总结[16]开始了对其中一个这样的描述符的研究。

### 14.5 进一步探索

文章[9,17]中包含了关于 MPEG 的详细回顾。在[18]中有关于 MPEG-4 中的自然音频编码的全面解释。[19]中介绍了结构化音频，[20]和[21]中有关于 MPEG-4 中的自然和结构化音频的详细介绍。

本章网站上的 Further Exploration 部分包含很多有用的链接：

- 非常全面的 MPEG 音频和 MP3 链接。

- MPEG 音频 FAQ。
- Fraunhofer-Gesellschaft 研究所的一篇非常出色的关于大步长可伸缩性工具的文献，“MPEG 4 Audio Scalable Profile”。它允许解码器根据有效带宽决定使用多少工具和在何种复杂度进行解码。

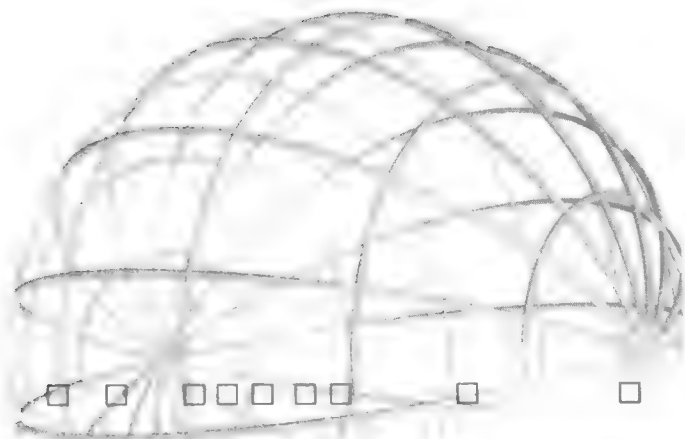
## 14.6 练习

- (a) 依照式 (14.1)，在 1000Hz 处安静的阈值是多少？  
(回想一下，这个等式是在 0dB 上使用 2kHz 作为参考。)
- (b) 推导式 (14.1) 并且令它为零，求出使曲线最小的频率。我们最敏感的频率是多少？提示：需要计算才能得到答案。
- 响度与振幅。60dB 的 1000Hz 声音和 60dB 的 100Hz 声音，哪一个更响？
- 在图 14-1 中，对于 Fletcher-Munson 曲线（新版本的），观察这些数据的方法是，设定 y 轴的值、声压值，然后测量人类的有效感知响度估计。给出一组观察值，我们必须怎样做才能把它们转换为图中所示的接收响度曲线？
- 两个音色同时产生。假设音色 1 已经确定，而音色 2 的频率可以变化。音色 1 的临界带宽是音色 2 的频率范围，在此我们可以听到敲打声和一个难听的声音。敲打声是一种比两种接近音色更低频率的泛音；它们因为两个音色的频率差异而产生。临界带宽在一个频率范围之内，超过这个范围，两个音色的声音就有不同的音调了。
  - 粗略估计 220Hz 的临界带宽是多少？
  - 说明如何设计一个实验来测量临界带宽？
- 在 Web 上查找下面心理声学现象的含义：
  - 虚拟音高
  - 听觉情景分析
  - 八度音程有关的复杂音色
  - 三全音怪论
  - 不和谐的复杂音色
- 如果在 MPEG 音频第一层的采样频率  $f_s$  是 32ksps，那么每一个 32 个子带中的子带带宽是多大？
- 假定第 8 个频段的遮掩音色是 60dB，那么在它停止 10 毫秒之后，第 9 个频段的遮掩效果是 25dB。
  - 如果第 9 个频段的原始信号是 40dB，那么 MP3 会如何？
  - 如果原始信号是 20dB 呢？
  - 针对 (a)、(b) 两种情况，应该给第 9 个频段分配多少位？
- 为了结合时间遮掩，MPEG 的第三层 (MP3) 音频与第一层有什么不同？
- 用作的理解向音频设备的销售员解释 MP3。
- 实现 MDCT，只对一个有 36 个样本信号，比较和 DCT 得到的频率的不同。当声音频率较低时，哪种方法能更好把能量集中到前面几个系数？
- 把一段 CD 音频转换为 MP3。比较原始音频质量和压缩后的版本——你能够听出区别吗？（多数人不能）
- 对于两通道立体声，我们一般都通常采用第二个通道和第一个并行的方式，利用从第一个通道收集的信息来压缩第二个通道。讨论你对实现过程的看法。

## 14.7 参考文献

- [1] D.W. Robinson and R.S. Dadson, "A Re-determination of the Equal-Loudness Relations for Pure Tones," *British Journal of Applied Physics*, 7: 166–181, 1956.
- [2] H. Fletcher and W.A. Munson, "Loudness, Its Definition, Measurement and Calculation," *J. of the Acoustic Society of America*, 5: 82–107, 1933.
- [3] T. Painter and A. Spanias, "Perceptual Coding of Digital Audio," *Proceedings of the IEEE*, 88(4): 451–513, 2000.
- [4] B. Truax, *Handbook for Acoustic Ecology*, 2nd ed. Burnaby, BC, Canada: Cambridge Street Publishing, 1999.
- [5] D. O'Shaughnessy, *Speech Communications: Human and Machine*, Los Alamitos, CA: IEEE Press, 2000.
- [6] A.J.M. Houtsma, "Psychophysics and Modern Digital Audio Technology," *Philips Journal of Research*, 47: 3–14, 1992.
- [7] E. Zwicker and U. Tilmann, "Psychoacoustics: Matching Signals to the Final Receiver," *Journal of the Audio Engineering Society*, 39: 115–126, 1991.
- [8] D. Lubman, "Objective Metrics for Characterizing Automotive Interior Sound Quality," in *Inter-Noise '92*, 1067–1072.
- [9] D. Pan, "A Tutorial on MPEG/Audio Compression," *IEEE Multimedia*, 2(2): 60–74, 1995.
- [10] P. Noll, "MPEG Digital Audio Coding," *IEEE Signal Processing Magazine*, 14(5): 59–81, Sep. 1997.
- [11] *Information Technology — Generic Coding of Moving Pictures and Associated Audio Information, Part 7: Advanced Audio Coding (AAC)*, International Standard: ISO/IEC 13818-7, 1997.
- [12] *Information Technology — Coding of Audio-Visual Objects, Part 3: Audio*, International Standard: ISO/IEC 14496-3, 1998.
- [13] *Information Technology — Multimedia Framework*, International Standard: ISO/IEC 21000, Parts 1-7, 2003.
- [14] *Information Technology — Multimedia Content Description Interface, Part 4: Audio*, International Standard: ISO/IEC 15938-4, 2001.
- [15] A.T. Lindsay, S. Srinivasan, J.P.A. Charlesworth, P. N. Garner, and W. Kriechbaum, "Representation and Linking Mechanisms for Audio in MPEG-7," *Signal Processing: Image Communication*, 16: 193–209, 2000.
- [16] P. Philippe, "Low-Level Musical Descriptors for MPEG-7," *Signal Processing: Image Communication*, 16: 181–191, 2000.
- [17] S. Shlien, "Guide to MPEG-1 Audio Standard," *IEEE Transactions on Broadcasting*, 40: 206–218, 1994.
- [18] K. Brandenburg, O. Kunz, and A. Sugiyama, "MPEG-4 Natural Audio Coding," *Signal Processing: Image Communication*, 15: 423–444, 2000.
- [19] E.D. Scheirer, "Structured Audio and Effects Processing in the MPEG-4 Multimedia Standard," *Multimedia Systems*, 7: 11–22, 1999.
- [20] J.D. Johnston, S.R. Quackenbush, J. Herre, and B. Grill, "Review of MPEG-4 General Audio Coding," in *Multimedia Systems, Standards, and Networks*, ed. A. Puri and T. Chen, New York: Marcel Dekker, 2000, 131–155.
- [21] E.D. Scheirer, Y. Lee, and J.W. Yang, "Synthetic Audio and SNHC Audio in MPEG-4," in *Multimedia Systems, Standards, and Networks*, ed. A. Puri and T. Chen, New York: Marcel Dekker, 2000, 157–177.





## 第三部分 多媒体通信和检索

### 第 15 章 计算机和多媒体网络

### 第 16 章 多媒体网络通信和应用

### 第 17 章 无线网络

### 第 18 章 数字图书馆中基于内容的检索

多媒体对网络和系统提出了巨大的需求。在这一部分中，我们将考察几个基本的和具有挑战性的重要多媒体网络和应用。

#### 多媒体网络

随着光纤技术的突破，网络带宽不断增长，我们正在目睹电信网络、计算机和多媒体网络的集成，以及混合通信类型（Internet 电话、视频点播等）的浪潮。多路复用和调度技术也正在被反复研究。而且，我们正在目睹无线网络的兴起（想想我们的移动电话和 PDA）。

在第 15 章中，我们将讨论计算机和多媒体网络的基本问题和技术，第 16 章将继续介绍多媒体网络通信和应用。第 17 章将简要介绍无线网络，以及这些网络上和多媒体通信有关的问题。

#### 数字图书馆中基于内容的检索

从多媒体数据库中自动检索在句法和语义上有意义的内容极其困难，尤其是当多媒体数据库内容已经极为丰富且数据库的规模也在急剧增长的情况下。第 18 章将讨论多媒体数据库系统的这一特定应用，探讨与在数字图书馆中进行基于内容的检索、存储和浏览有关的问题。

## 第 15 章 计算机和多媒体网络

我们熟悉和日益依赖的计算机网络对于现代计算环境而言是不可或缺的。多媒体网络面临的主要问题和使用的技术与计算机网络一样。此外，持续增长的对各种多媒体通信的需求使得网络成为最热门的研究和发展领域之一。

本章将先介绍一些计算机和多媒体网络中常用的技术和方法，接下来再介绍各种高速网络，因为它们正逐渐成为当前多媒体系统的核心部分。

### 15.1 计算机和多媒体网络基础

#### 15.1.1 OSI 网络的层次

一直以来，网络通信被认为是一件涉及不同层次协议的复杂任务。因此，国际标准化组织（ISO）于 1984 年提出了一种叫做开放式系统互连（Open Systems Interconnection, OSI）的多层次协议体系结构，这个体系最终标准化为 ISO 7498。OSI 参考模型的网络层如下所示[1, 2]：

1) **物理层**。该层定义物理接口的电气与机械属性（如信号强度，连接器的规范等），还指明了物理接口电路的功能与过程时序。

2) **数据链路层**。该层指定建立、维护和终止连接的方法，如传送和同步数据帧、差错修复和访问物理层的协议。

3) **网络层**。该层定义数据在网络上由一端到另一端的路由，比如电路交换和包交换。提供寻址、网际互连、错误处理、拥塞控制和包排序等服务。

4) **传输层**。该层在支持终端用户应用程序或服务的终端系统之间提供端到端的通信，支持面向连接或者无连接的协议，提供错误修复和流量控制功能。

5) **会话层**。该层协调不同主机上用户应用程序之间的交互，管理会话（连接），比如长文件传输的完成。

6) **表示层**。该层处理传输的数据的语法，比如不同数据格式的转换和针对于不同惯例的编码、压缩和加密。

7) **应用层**。该层包含各种应用程序和协议，比如 FTP、Telnet、HTTP、SNMP、SMTP/MIME 等。

#### 15.1.2 TCP/IP 协议

虽然 OSI 协议体系有助于发展计算机网络，但是由于一种叫做 TCP/IP 的更为实用的协议族的竞争，它并没有得到广泛接受。TCP/IP 协议族在美国国防部的资助下先于 OSI 开发出来。由于在因特网上得到广泛应用，TCP/IP 协议族已经成为事实上的标准。

图 15-1 比较了 OSI 和 TCP/IP 协议的体系结构。可以看出，TCP/IP 缩减了协议的层数，而且把 OSI 体系的最上面三层合并为一个应用层。实际上，TCP/IP 非常灵活，以至于有的时候应用层的协议可以直接操作 IP。

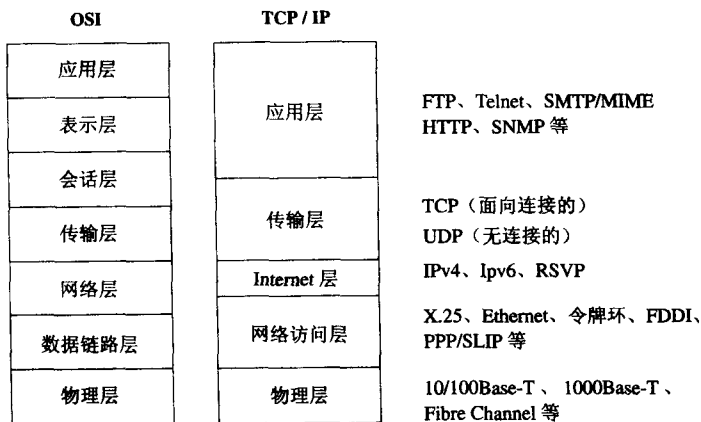


图 15-1 OSI 和 TCP/IP 协议的体系结构和示例协议的比较

### 1. 传输层：TCP 和 UDP

TCP 和 UDP 是 TCP/IP 中用来实现端到端（点对点）通信的两种传输层协议。

422

#### （1）传输控制协议（Transmission Control Protocol, TCP）

TCP 是面向连接的，因为它提供网络上通信进程间可靠的数据传输。它处理将应用数据发送到目的进程的过程，但并不关心数据报或者包的大小。然而，TCP/IP 是为包交换网络建立的。因此，由于并没有连接电路，数据仍然需要进行分包。

TCP 依靠 IP 层来将消息发送到 IP 地址所确定的目标计算机上。它支持消息包装、错误检测、重传、包重排序和多路复用。由于运行 TCP/IP 的进程需要与一个远程进程建立多个网络连接，因此多路复用这个功能通过用端口号标识每个连接的方式来实现。

对于每个 TCP 连接，进行通信的计算机都分配一个叫做窗口的缓冲区来接收和发送数据。流量控制通过保证发送的数据不会从目的计算机的窗口中溢出来实现。这样，同时可以传输的最大数据量就由两台计算机中窗口较小者确定。

每个 TCP 数据报的头包含源端口与目标端口、序号、校验和、窗口字段、确认号和其他字段。

- 源端口与目标端口对于源进程来说，用来确定将向何处发送消息，而对于目标进程来说，它们被用来确定将向何处回复消息（地址在 IP 层中指定）。
- 当包在网络上传输时，它们可能是无序到达的（经过不同的传送路径），也可能会丢失，甚至重复。通过顺序号就可以对到达的包重新进行排序，检测是否有丢包的情况发生。顺序号标识的实际上是每个包的第一个字节的计数，而不是对应整个包的序列号。
- 校验和用于确认包到达的时候是否被破坏而不管通道的干扰，这种校验的确定度是很高的。如果接收到的包计算得到的校验和与传送过来的校验和不匹配，那么这个包将会被丢弃。
- 窗口字段指明当前计算机的缓冲区最多可以接收多少字节的数据。它通常随确认包一同传送。
- 确认（ACK）包有一个 ACK 号，这个编号指明当前顺序上已经正确接受到的字节数（对应于第一个丢失的包的顺序号）。

源进程向目的进程发送的数据报的大小由窗口数决定，在发送更多数据之前，它必须等待确认（ACK）。ACK 包到达的时候同时，也包含一个新的窗口大小信息，通过这个信息，源进程可以知道目标缓冲区还可以接收多少数据。如果没有在重传超时（RTO）所规定的时间内收到 ACK，源进程将从本地窗口缓冲区中重新发送相应的包。TCP/IP 并没有指定拥塞控制机制，然而实现

TCP/IP 的时候必须包含这一机制。

虽然 TCP 是可靠的,但对于许多实时多媒体应用,比如视频流来说,重传的开销是非常大的。因此这些应用一般使用 UDP。

## (2) 用户数据报协议 (User Datagram Protocol, UDP)

UDP 是无连接的协议,因为被传送的消息是一个数据报。如果消息过长或者需要有担保的投递,那么这些工作要由应用层来完成。从本质上说,UDP 仅仅提供了多路复用和通过校验和实现的错误检测。虽然 UDP 的头中有指明源端口号和目的端口号的相应字段,但是源端口号实际上是可选的,因为目的计算机并不需要回复消息 (UDP 中没有确认)。

虽然 UDP 中数据的传输速度高于 TCP,但是这种传输是不可靠的,特别是在一个拥塞的网络环境中。光纤网络品质的不断提高使丢包率不断减少。在大多数实时多媒体应用中 (比如视频流和音频流),迟到的包只是被丢弃。虽然高层的协议可以控制重新传输,流量控制和拥塞避免,但是为了有可接受的服务质量 (QoS),我们仍然需要开发更多实用的错误隐蔽 (error concealment) 技术。

## 2. 网络层: 网际协议

IP 层提供两个基本的服务: 包寻址和包分割。在任何局域网 (Local Area Network, LAN) 内,点对点的消息传送都可以很容易地实现,实际上,LAN 还常常支持广播。然而,如果要将消息发送到另一个 LAN 中的计算机上的时候,就需要一个中介设备来转发消息了。IP 协议提供在所有互连的网络中对计算机进行全局寻址的功能,这里,每一个入网的计算机 (或设备) 都被分配了一个全局唯一的 IP 地址。

为了使 IP 包能够在不同的 LAN 或者广域网 (Wide Area Network, WAN) 中传输,我们需要使用网关或者路由器,它们利用路由表来指示消息传送到目的 IP 地址。网关是一台位于 LAN 边缘的计算机,它可以发送 IP 包到 LAN 网络接口和 WAN 网络接口,从而与其他不在这个 LAN 中的计算机进行通信。路由器就是一个可以接收包并把它们发送到同类网络中相应目的地址的设备。

IP 层还必须把收到包的地址翻译为合适的网络地址。另外,路由表指明包到达目的地址的最佳路由上的下一个路由器。由于最佳路由会随着节点的可用性、网络拥塞和其他因素而改变,路由器必须相互通信来决定当前 IP 组的最佳路由。这种相互的通信是通过互联网控制报文协议 (ICMP) 来实现的。

IP 是无连接的,它不提供端到端的流量控制。每个包都是独立处理,并不与过去或者未来的包发生联系。因此,包接收到的时候可能是无序的,也可能被丢弃或者重复。

当一个包要通过只能接收较小包的的网络的时候,包分割操作就必不可少。在这种情况下,IP 包会被分割为符合要求的较小的包,然后通过网络发送到下一个中继段,并在那里重新排序组装。

在 IP 当前的版本,也就是 IPv4 (IP 版本 4) 中,IP 地址是一个 32 位的数字,常常用点分十进制方式表示 (比如,128.77.149.63 = 10000000 01001101 10010101 00111111)。这种 32 位寻址理论上拥有  $2^{32} \approx 40$  亿个地址,看起来似乎远远超过需要。但实际上,我们将很快用完这些地址 (预计在 2008 年)。

出现这种情况并不仅仅因为个人电脑和无线设备的迅猛发展,IP 地址分配过于浪费也是一大原因。比如说,IP 地址是有一定格式的 (网络号,主机号)。对于很多网络号,其下使用的主机号的比例都相对较小,更不用说一些非活动的主机仍然占用着它们先前分配到的地址。

作为一种应对可用 IP 地址短缺 (由于服务提供商的限制或者成本因素) 的短期解决方法,一些 LAN 使用代理服务器或者代理服务器实现的网络地址翻译化 (Network Address Translation, NAT) 功能 (还有内容缓存或者其他特性)。NAT 设备把 LAN 从互连网络中分离出来,而且只需要一个 IP 地址就可以完成 LAN 中所有计算机与外部的通信。LAN 中每一个计算机都被分配一个



本地 IP 地址, 而外部连接的网络是无法访问这个地址的。NAT 设备常常维护一个动态 NAT 表, 这张表用于把外部 IP 地址所用的通信端口转化为相应进行通信的计算机的内部 IP 地址。

当本地计算机使用本地 IP 地址发送一个 IP 包时, 这个包将会经过 NAT 设备, 由 NAT 把源 IP 地址转换为 NAT 设备对应的全局 IP 地址。当一个 IP 包到达 NAT 设备的某个通信端口的时候, 目的地址将会按照 NAT 表转换为本地 IP 地址, 这个包就可以被转发到正确的计算机上了。

1995 年 1 月, IPv6 (IP 版本 6) 由因特网工程任务组 (Internet Engineering Task Force, IETF) 推荐为下一代 IP (IPng), 记录 RFC1752——“关于下一代 IP 协议的推荐标准”中。IPv6 对 IPv4 进行了大量改进, 采用 128 位的地址, 一共允许  $2^{128} \approx 3.4 \times 10^{38}$  个地址[2]。这必然能够在很长一段时间内 (不是永远) 解决 IP 地址短缺的问题。

## 15.2 多路复用技术

现代通信链接通常拥有很高的容量, 特别是当光纤网络出现以后更是如此。当链接容量远远超出任何一个用户的传输速率时, 为了共享容量, 必须引入多路复用技术。

在这一节内, 我们先分析基本的多路复用技术, 接下来再对 ISDN、SONET 和 ADSL 等现代网络进行综述。

### 15.2.1 多路复用技术基础

#### 1. 频分多路复用 (Frequency Division Multiplexing, FDM)

在 FDM 中, 多个通道按照它们的频率来组织。打个比方, 广播和电视就是使用 FDM 的很好的例子——它们通过把频带分为多个通道来共用有限的广播频带带宽。目前, 有线电视与 FDM 数据网络甚至更为类似, 因为它们传输的媒介是相似的。对于普通的声音通道和电视通道, 一般来说, 声音的带宽为 4kHz, NTSC TV 的带宽为 6MHz, PAL 或者 SECAM TV 的带宽为 8MHz。

425

为了使 FDM 正常工作, 模拟信号必须先按照每个通道独有的载波频率  $f_c$  进行调制。这样, 信号将会占用一个以  $f_c$  为中心的带宽  $B_s$ 。接收器用一个已经按照通道要求调谐好的带通滤波器来获取信号, 然后再使用解调器来对信号进行解码。

基本的调制技术包括调幅 (Amplitude Modulation, AM)、调频 (FM) 和调相 (PM)。调频与调相组合起来就是当前很多应用中所使用的正交幅度调制 (QAM) 技术[1, 2]。

数字数据常常使用模拟信号来传输。最典型的例子就是用于在电话网络上传输数字数据的调制解调器。数字数据被调制为载波信号用于传输, 接收器再对信号进行解调来恢复成数字数据。基本的调制技术包括幅移键控 (ASK)、频移键控 (FSK) 和相移键控 (PSK)。正交相移键控 (QPSK) 是 PSK 的一种高级版本, 它使用  $90^\circ$  的相移而不是  $180^\circ$  的相移[2]。和 QAM 一样, 它也可以把相位和幅度组合起来, 这样就可以在每个副载波频率上传送多个位了。

#### 2. 波分多路复用 (Wavelength Division Multiplexing, WDM)

WDM 是 FDM 的一种变形, 它在光纤传输中特别有用。从本质上说, 光线用不同波长代表通道。在源端, 光线被组合起来并在同一根光纤内传输; 在接收端, 光线又重新被拆分。光线的组合和拆分是由光纤设备 (如 ADM) 来完成的, 光纤设备的可靠性和效率都高于普通电路。由于每条光纤的带宽都非常高 (每个频带大于 25 万亿赫兹), 所以 WDM 的容量是巨大的, 具有大量可复用的通道。这样一来, 光纤主干总计的码率将有可能达到数万亿比特每秒。

WDM 有两种变型:

- 稠密波分多路复用 (Dense WDM, DWDM) 使用了间隔较密的波长, 因而比 WDM 可以使用更多的通道 (例如, 大于 32 个)。

- 宽带波分多路复用 (Wideband WDM, WWDM) 它允许波长范围较大的彩色光 (例如, 长波段的 1310~1557nm, 短波范围的 850nm) 的传输, 从而获得比 WDM 更大的通道容量。

### 3. 时分多路复用 (Time Division Multiplexing, TDM)

前面描述过, FDM 更适合模拟信号, 对于数字计算机网络就不太适用了。而 TDM 就是一种直接多路复用数字数据的技术。如果源数据是模拟的, 它必须先进行数字化并转化为脉冲编码调制 (PCM) 样本 (如第 6 章所述)。

在 TDM 中, 多路复用是在时间维 ( $t$ ) 上进行的。对  $m$  ( $m > 1$ ) 个通道, 需要使用多个缓冲区。在一帧数据形成之前, 在  $m$  个循环的每一个时隙中都会从一个缓冲区中取出一位 (或者一字节)。TDM 帧形成后将会被传输, 然后在接收端重新分离。

426

上面描述的方案称作同步 TDM, 因为在这种方案中, 每个缓冲区都被依次扫描并同等对待。如果在某一个时隙, 某些源 (相应的缓冲区) 没有数据要传输, 这个时隙就浪费了。异步 TDM (或者统计 TDM) 根据这种情况来收集各个缓冲区的信息。它只在  $k$  ( $k < m$ ) 个时隙中去扫描  $k$  个可能会有数据发送的缓冲区。在载波数据率相同的情况下, 异步 TDM 可以达到更高的吞吐量。不过, 由于源地址同时也必须和数据一起传输, 使分离能够正确进行, 因此这种方法是有一定的额外开销的。

一般来说, 电话通道上的语音数据的带宽为 4kHz。按照奈奎斯特定理, 为了使数字化效果更好, 每秒需要有 8000 个样本。这样, 每个样本间的间隔为  $125\mu\text{s}$ 。每一个通道每个样本可以传送 8 位数据, 这样每个语音通道传输数据率 (包括数据和控制位) 为  $8 \times 8000 = 64 \text{ kbps}$ 。

在北美和日本, T1 载波<sup>①</sup>就是一种有 24 个语音通道 (即 24 个时隙) 的同步 TDM, 其中 23 个通道用于传送数据, 最后 1 个通道用于同步。每个 T1 帧有  $8 \times 24 = 192$  位, 外加 1 位用于分割帧 [1, 2]。这样, 总计的数据传输率就是每  $125\mu\text{s}$  193 位, 也就是  $193 \text{ 位/样本} \times 8000 \text{ 样本/秒} = 1.544 \text{ Mbps}$ 。

4 个 T1 载波还可以进一步复用生成 T2。注意, T2 总的数据传输率为  $6.312 \text{ Mbps}$ , 大于  $4 \times 1.544 = 6.176 \text{ Mbps}$ , 这是因为 T2 需要更多的分割位和控制位。按照同样的原理, 可以构造 T3 和 T4。

ITU-T 定义了类似的载波格式, 第一级 (E1) 从  $2.048 \text{ Mbps}$  开始, 这一级里每一帧由 32 个时隙组成:  $8 \times 32 \times 8000 = 2.048 \text{ Mbps}$ 。在 32 个时隙中, 2 个用于分割帧和同步帧, 其余 30 个是数据通道。每高一级, 如 E2, E3 等等, 复用的通道数变为原来的 4 倍。表 15-1 比较了两种 TDM 载波标准。

表 15-1 TDM 载波标准的比较

格式	通道数	数据率 (Mbps)	格式	通道数	数据率 (Mbps)
T1	24	1.544	E1	32	2.048
T2	96	6.312	E2	128	8.448
T3	672	44.736	E3	512	34.368
T4	4032	274.176	E4	2048	139.264
			E5	8192	565.148

## 15.2.2 综合业务数字网络

一个世纪以来, 用于模拟声音传输的公共交换式电话系统一直支持普通老式电话服务 (Plain

① T1 载波的格式称为 DS1, T2 的载波格式称为 DS2, 以此类推。非严格地说, 这两种表示方法 (T 和 DS) 通常可以互换。

Old Telephone Service, ROTS)。在20世纪80年代,ITU-T开始开发ISDN(Integrated Services Digital Network, 综合业务数字网络)来满足各种数字化服务的需要(例如,呼叫者ID,建立即时呼叫,远程会议),在这些服务中,数字数据、语音,甚至视频(例如在视频会议中)都能够传输。

427

默认情况下,ISDN指窄带ISDN。ITU-T随后开发了宽带ISDN(B-ISDN)。它默认的交换技术是异步传输模式(Asynchronous Transfer Mode, ATM)[3],稍后对其进行讨论。

ISDN定义了多种全双工通道类型:

- **B通道** 每个通道64kbps。B通道用于数据传输。大多数情况下,它们是电路切换的,但是它们也支持包交换。如果需要的话,B通道可以很容易地替代POTS。
- **D通道** 16kbps或者64kbps。D通道关心的是呼叫建立、呼叫控制(呼叫转发、呼叫等待等),以及网络维护。拥有一个D通道的好处是可以在B通道传输数据时实时地在D通道内进行控制和维护。

接下来看一下ISDN的主要规范:

- 它采用同步TDM,其中上面提到的通道被复用。
- 根据数据和订阅的速率,有两种接口可以供用户使用。
  - **基本速率接口**提供两个B通道和一个D通道(16kbps),共有144kbps( $64 \times 2 + 16$ )被复用,并在一个192kbps的链路上传输。
  - **初级速率接口**在北美和日本使用的版本提供23个B通道和一个D通道(64kbps),而在欧洲使用的版本则提供30个B通道和2个D通道(64kbps)。23个B和1个D刚好就是T1,因为T1有24个时隙,并且数据传输率为 $24 \times 64\text{kbps} \approx 1\,544\text{Mbps}$ ;而30个B和2个D就对应于E1,E1有32个时隙(其中30个为用户通道),并且数据传输率为 $32 \times 64 = 2.048\text{kbps}$ 。

由于窄带ISDN的传输率较低且成本较高,因此它并不能满足数据与多媒体网络的需要。对于某些家用电脑或者Internet用户来说,它已经被接下来要讨论的电缆调制解调器和非对称数字用户线(ADSL)所取代。

### 15.2.3 同步光纤网络

同步光纤网络(Synchronous Optical Network, SONET)最初是一个Bellcore开发的标准,当时是为了使光纤支持比T3更高的数据传输率。后来,SONET标准由ANSI调整并批准,作为ANSI T1.105, T1.106和T1.107。SONET使用电路切换和同步TDM。

428

在光纤网络中,电子信号必须转换为光信号才能进行传输,在接收端再转换回来。相应地,SONET对电子信号使用同步传输信号(Synchronous Transport Signal, STS),对光信号使用光载波器(Optical Carrier, OC)。

一个STS-T(OC-1)帧含有810个TDM字节。每个帧传输时间为 $125\mu\text{s}$ ,也就是每秒8000帧,所以数据传输率就是 $810 \times 8 \times 8\,000 = 51.84\text{Mbps}$ 。其他的STS-N(OC-N)信号都是STS-1(OC-1)信号的进一步复用。例如,三个复用的STS-1(OC-1)信号,为每个STS-3(OC-3)一是 $155.52\text{Mbps}$ 。

ITU-T并没有采用SONET,而是开发了一种称为同步数字体系(SDH)的类似标准。SDH使用同步传输模块(STM)技术。STM-1是SDH中级别最低的,它对应于SONET中的STS-3(OC-3)。

表15-2列出了SONET的电子和光学级别以及SDH中对应的级别和传输率。在所有列出的级别中,OC-3(STM-1)、OC-12(STM-4)、OC-48(STM-16)和OC-192(STM-64)是最常使用的。

表 15-2 SONET 和 SDH 中的等价级别

SONET 电子级	SONET 光学级	SDH 对应	线率 (Mbps)	有效载荷率 (Mbps)
STS-1	OC-1	—	51.84	50.112
STS-3	OC-3	STM-1	155.52	150.336
STS-9	OC-9	STM-3	466.56	451.008
STS-12	OC-12	STM-4	622.08	601.344
STS-18	OC-18	STM-6	933.12	902.016
STS-24	OC-24	STM-8	1244.16	1202.688
STS-36	OC-36	STM-12	1866.24	1804.032
STS-48	OC-48	STM-16	2488.32	2405.376
STS-96	OC-96	STM-32	4976.64	4810.752
STS-192	OC-192	STM-64	9953.28	9621.504

15.2.4 非对称数字用户线路

非对称数字用户线路 (Asymmetric Digital Subscriber Line, ADSL) 是电话工业界对最后一公里 (为每个家庭提供快速网络服务) 挑战的回应。它采用了相对较高的下行数据传输率 (从网络到用户) 和较低的上行数据传输率 (从用户到网络), 因此它是不对称的。

ADSL 利用已有的电话双绞线来传输正交幅度调制 (QAM) 数字信号。ADSL 线路上的信号带宽被提高到 1MHz 甚至更高, 而不是传统电话线上 4KHz 的音频信号。

ADSL 使用 FDM (频分多路复用) 来复用三个通道:

- 位于高端频谱的高速 (1.5~9Mbps) 下行通道。
- 中速 (16~640kbps) 的双工通道。
- 位于频谱低端<sup>①</sup> (接近 DC, 0~4kHz) 的 POTS 通道。

这 3 个通道都可以被划分为 4kHz 的子通道 (例如, 256 个子通道组成 1MHz 的下行通道)。这些子通道的复用方式依然是 FDM。

由于信号 (特别是接近或者就是 1MHz 的高频信号) 在双绞线上会很快衰减, 并且随着线路长度增加噪声不断增大, 在线路到达一定距离后, 信噪比将会跌落到不可接受的程度。表 15-3 给出了在不考虑桥接效应的前提下, ADSL 使用普通的铜芯双绞线时的距离限制情况。

ADSL 的关键技术是离散多音频 (Discrete Multi-Tone, DMT)。为了在可能存在噪声的通道内更好地传输 (上行或下行),

DMT 调制解调器先向每个子通道发送测试信号。然后就可以计算信噪比, 来动态决定向每个子通道发送的数据量。信噪比越高, 发送的数据越多。从理论上说, 256 个下行子通道 (每个子通道的传输速率都可以超过 60kbps) 的总的数据传输率将大于 15Mbps。实际上, 在目前的技术条件下, DMT 只能达到 1.5~9Mbps。

表 15-4 提供了各种数字用户线路 (xDSL) 的简要历史。DSL 对应于基本速率 ISDN 服务。HDSL 是在低带宽 (196kHz) 条件下达到 T1 (或者 E1) 的传输速率[2]。然而, 它需要两条双绞线来达到 1.544Mbps 的速率, 而达到 2.048Mbps 则需要三条双绞线。SDSL 在一条双绞线上提供

表 15-3 使用铜芯双绞线的 ADSL 的最大传输距离

数据率	线 宽	距 离
1.544Mbps	0.5mm	5.5km
1.544Mbps	0.4mm	4.6km
6.1Mbps	0.5mm	3.7km
6.1Mbps	0.4mm	2.7km

① 可以用一个 ISDN 通道替代低或中速的通道。

了与 HDSL 同样的服务。VDSL 是一个仍在发展中的标准，它是 xDSL 的未来形式。

表 15-4 数字用户线系统的历史

名 称	含 义	数 据 率	模 式
V.32 or V.34	声音频带调制解调	1.2 to 56kbps	双向
DSL	数字用户线	160kbps	双向
HDSL	高数据率数字用户线	1.544Mbps or 2.048Mbps	双向
SDSL	单线数字用户线	1.544Mbps or 2.048Mbps	双向
ADSL	不对称数字用户线	1.5 to 9Mbps	下行
		16 to 640kbps	上行
VDSL	超高数据率数字用户线	13 to 52Mbps	下行
		1.5 to 2.3Mbps	上行

## 15.3 LAN 和 WAN

局域网 (LAN) 局限在一个小的地理区域里面，通常只包含较少量的计算机。广域网 (WAN) 指跨越城市和国家的网络。在 LAN 和 WAN 中间，有时候会用到城域网 (Metropolitan Area Network, MAN) 的概念。

### 15.3.1 局域网

大多数 LAN 使用广播技术。它们都毫无例外地使用公用介质。因此，介质访问控制是一个重要的问题。

IEEE 802 委员会提出了 LAN 的 IEEE 802 参考模型。由于 OSI 参考模型的第 3 层及以上的层可以供 LAN、MAN 或 WAN 使用，因此 IEEE 802 主要面向底下的两层——物理层和数据链路层进行开发。特别地，数据链路层的功能有所增强，并且被划分为两个子层：

- **介质访问控制 (MAC) 层** 这一层在传输和接收的时候负责装配以及分解帧，执行寻址和错误修复，并控制对公用物理介质的访问。
- **逻辑链路控制 (LLC) 层** 这一层提供流量控制、错误控制和 MAC 层寻址。它同时扮演与高层之间接口的角色。LLC 在层次上要高于 MAC。

下面是一些现有的 IEEE 802 附属委员会和他们负责的领域：

- **802.1 (高层 LAN 协议)** 802.X 标准与 OSI 参考模型的关系，LAN 之间的互连和管理。
- **802.2 (LLC)** 逻辑链路控制 (LLC) 的一般标准。
- **802.3 (以太网)** 以太网的介质访问控制 (CSMA/CD) 和物理层规范。
- **802.5 (令牌环)** 令牌环的介质访问控制和物理层规范。
- **802.9** 介质访问控制和物理层上关于综合业务的 LAN 接口。
- **802.10 (安全性)** 能与其他 IEEE 802 标准共同使用的 LAN/MAN 安全性措施。
- **802.11 (无线 LAN)** 无线局域网 (WLAN) 的介质访问方法和物理层规范。
- **802.14 (基于有线电视的宽带通信网络)** 有线电视上多媒体服务的双向传输标准协议，如混合光纤同轴 (Hybrid FiberCoax, HFC) 电缆调制解调器和电缆网络。
- **802.15 (无线 PAN)** 无线个人区域网络 (WPAN) 的访问方法和物理层规范。个人区域网络 (PAN) 覆盖范围为 10 米这个数量级。
- **802.16 (宽带无线)** 宽带无线网络的访问方法和物理层规范。

### 1. 以太网

以太网是一个包交换的总线网络,它是目前最流行的 LAN。截止到 1998 年,以太网的普及率达到了 85%。发送消息的时候,接收者的以太网地址会附在消息上,而消息会被发送到总线上的所有站点上。只有目的计算机接收这个消息,而其他计算机将忽略该消息。

网络上的介质访问控制的问题可以由载波侦听多路访问/冲突检测(CSMA/CD)来解决。需要发送消息的站点首先要监听网络(载波侦听),一直到网络上没有传输为止。显然,有可能出现多个站点等待,然后同时发送消息的情况,这样就会出现冲突。在传输帧的时候,站点比较接收到的信号和发送的信号。如果它们不一样,则检测到冲突。一旦检测到冲突,这个站点就会停止发送这一帧,随机等待一段时间后重新传输。

以太网的一种非常好的传输介质是同轴电缆(或者新一代的光纤),但是也可以使用双绞线。因为它们其实就是电话线,在办公楼或者家中已经有这样的线,因而不需要重新安装。

有时候,人们会使用星型 LAN。在这种网络中,所有的站点都直接连接到一个集线器(hub)上,它可以帮助解决传输质量低的问题。Hub 是一个活跃的设备,扮演中继器的角色。每当它从一个站点接收到信号,就将其复制,以使其他的站点都能够接收。从逻辑上说,它仍然是一个总线型网络,虽然物理上仍是一个星型网络。

432

普通的以太网的最大传输速率为 19Mbps。对于 10Mbps 的 LAN,10BASE-T 使用的无屏蔽的双绞线最大有效距离为 100m,而 10BASE-F 中使用光纤的最大有效距离可以达到 2 千米。

快速以太网(也称为 100BASE-T)的最大传输率为 100Mbps<sup>①</sup>,并与以太网完全兼容。确实,通过交换机(不是 Hub)混用 100BASE-T 和 10BASE-T 实在是太普遍了。也就是说,服务器和 100BASE-T 的交换机之间连接 100BASE-T 链路,而交换机和 workstation 之间是多个 10BASE-T 链路。由于交换机能够同时处理多个通信,因此所有的工作站最高都能够以 10 Mbps 的速率进行通信。

### 2. 令牌环

顾名思义,令牌环上的站点连接成一个环型拓扑结构。数据帧按照同一个方向在环上传输,并可以被所有站点读取。如果站点实际上(物理上)都连接到一个 Hub 上,那么这种环状结构就只是逻辑上的存在了,这里 Hub 会重复并中继“环”上的信号。

在这个环空闲的时候,一个叫做令牌的小帧将在环上循环传播。一个站点 S 要想开始传输,则必须等到令牌到达。源站点 S 这时就会占有这个令牌,并将它转化为数据帧的前端,接下来这个帧就会在环上传输并被目的站点接收。数据帧会继续在环上传输直到返回源站点 S 为止, S 将令牌释放并把它重新放回到环上。

我们通过只允许一个令牌来控制对共享介质的访问,因此,这里不会出现冲突。在默认情况下,环体现了转盘的思想。每当令牌被释放,下一个站点就会得到它,以此类推。此外,也可以应用多优先级策略来控制共享访问,即给定优先级的站点如果可以获取与其相当或者较低优先级的令牌,则可以开始传输帧;否则,它只能预约并等待。

使用非屏蔽双绞线的令牌环的传输速率为 4Mbps 或者 16Mbps。4Mbps 的令牌环按照上述方法来管理令牌。在 16Mbps 的令牌环中,源站点一发送出数据帧,就会释放令牌。这种方法通过让多个帧同时在环上传输来提高环的利用率。目前已经有新的技术实现了 100Mbps 的令牌环[2],而 1998 年的 IEEE 802.5v 则是千兆令牌环的一种可行性研究。

### 3. 光纤分布式数据接口(Fiber Distributed Data Interface, FDDI)

FDDI 是从最初的令牌环基础上发展而来。FDDI 的介质访问控制(MAC)与 IEEE 802.5 中为令牌环规定的 MAC 方法十分相似。

① 下一代的以太网将是千兆以太网和 10G 以太网,这些内容将在后面讨论。

FDDI 是双星型拓扑结构, 其中主环用于数据传输, 而辅助环用来容错[5]。如果两个环中都检测到错误, 它们能够合并起来像一个环一样工作。

FDDI 的码率为 100Mbps。由于它的传输率相对较快, 所以源站点在发送数据帧之前只是将令牌吸收 (而不是像在最初的令牌环中那样将令牌转换为数据帧的一部分)。

在 FDDI 中, 一个站点一旦获取了令牌, 它会获得一段时间, 在这段时间内它可以尽量多地发送数据帧。而且, 一旦数据帧发送完毕, 令牌就会被释放 (提前令牌释放)。

433

FDDI 网络最远可以跨越 100km 的距离。它最多可以支持 500 个站点, 只要临近站点的距离不超过 2 千米即可。因此, FDDI 主要作为 LAN 或者 MAN 的骨干网。

FDDI 既支持同步模式也支持异步模式[5]。同步模式下可以做到带宽预留, 并保证数据传输达到同步容量。异步模式与令牌环协议比较相似。FDDI-2 还另外支持一种模式——等时模式[5], 这种模式下, 网络是分时的, 每台计算机都能得到一个固定的时间片。这样, FDDI 就能为延迟敏感的应用 (比如音频和视频) 提供等时服务, 而对网络中的其他应用提供同步和异步服务了。

### 15.3.2 广域网

广域网 (WAN) 通常指跨越城市和国家之间的网络。它们都使用了某种类型的交换技术, 而不是广播。

#### 交换技术

电路交换和包交换是两种常用的交换技术。而后者还有帧中继和信元中继两种变体。

- **电路交换** 公共交换电话网络 (PSTN) 就是电路交换的一个例子, 在这个网络中, 必须建立一个端到端的电路 (这种情况下是双工) 来用于保证连接期间的带宽。虽然这个网络最初是用于语音通信的, 但是它也可以用于数据传输。它是窄带 ISDN 的基础 (参见 15.2.2 节)。为了解决多用户和可变的传输率的问题, 它采用 FDM 或者同步 TDM 复用。

如果用户需要较固定的数据率连接, 如恒定速率的视频传输, 则电路交换就有优势。但对一般意义上的多媒体传输, 特别是可变 (有时还是突变) 数据率的情况下, 它的效率就比较低。

- **包交换** 几乎所有的数据网络都使用包交换技术, 在这些网络中, 传输率常常是变化甚至猝发的。在传输前, 数据被分成一些小的包, 通常为 1000 字节或者更少。每个包头都携带必要的控制信息, 比如目的地址、路由等。包交换最常使用的协议是 X.25。

通常有两种方法用来完成包交换和建立路由: 数据报和虚拟电路。在数据报中, 每个包都单独作为一个数据报。但因为传输前并不规定好路由, 因此包可能会丢失或者乱序到达。

就像在 TCP/IP 中一样, 由接收站点负责检测和修复错误。

在虚拟电路中, 路由通过路由上所有节点的请求和接收来预先决定。之所以称为“电路”, 是因为路由是固定的 (一旦协商好), 并在整个传输过程中不变; 而它又是“虚拟的”, 是因为这个“电路”只是逻辑上的而不是专用的, 并且从同一个源传输到同一个目的地的包可以通过不同的“电路”来实现。虚拟电路中序列化 (包排序) 相对比较容易。检测到错误的时候通常会要求重传。

434

包交换在网络比较拥塞的时候效率比较低, 并且会因为大量包延迟时间过长和丢失而使它变得很不可靠。

- **帧中继** 现代高速链路的错误率很低。在光纤中, 错误率可以低至  $10^{-12}$  这个数量级。这样一来, 就没有必要像普通包交换技术 (X.25) 那样, 在每个包中添加许多位以用于错误检测。与 X.25 一样, 帧中继在数据链路控制层工作。帧中继对 X.25 作出了如下改变:

- **错误检测的削减** 没有更多的确认, 没有更多的跳到跳的流量控制和错误控制。而且, 端到端的流量控制和错误控制可以在高层实现。
- **层数减少** 多路复用和交换式虚拟电路从 X.25 中的第三层改到了第二层。X.25 中的第三层被去掉了。

帧中继基本上是包交换的一个简化版本，只提供了最少的服务。帧长度最长可达到 1 600 字节。如果接收到坏帧，则将其丢弃。因此，帧中继的传输率大幅提高，位于 T1 (1.5Mbps) 和 T3 (44.7Mbps) 之间。

- 信元中继 (ATM) 非同步传输模式采用长度固定并且比较小的包，这种包称为信元。因此，ATM 也称为信元中继。

如图 15-2 所示，较小的包有助于减少 ATM 网络中的延迟时间。在图 15-2a 中，当一个黑色的包紧跟着一个正常大小的包 (如 1KB) 后面到达，它必须等待前一个包传输完成，这就引起了序列化延迟。如果包 (信元) 比较小，如图 15-2b 所示，黑色的信元需要等待的时间就会缩短。这样可以显著提高网络的吞吐量，特别有利于实时多媒体应用。ATM 被认为有潜力达到几百 (几千) Mbps 的传输率。

图 15-3 从码率和复杂度角度比较了上述四种交换技术。从图中可以看出，电路交换最简单，并且提供了持续 (固定) 的传输率，而包交换恰恰相反。

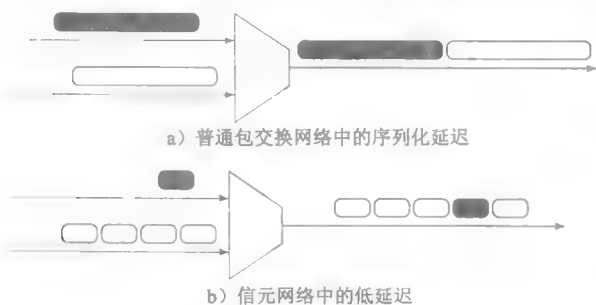


图 15-2 等待时间

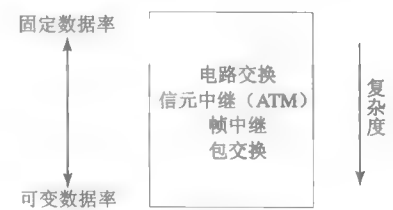


图 15-3 不同交换技术之间的比较

15.3.3 异步传输模式

从 20 世纪 80 年代开始，数据通信和多媒体服务 (语音、视频等) 的迅速增长给电信网络带来了极大挑战。由光纤带来的带宽增加，使得宽带 ISDN (B-ISDN) 成为事实。到 1990 年，ITU-T (原来的 CCITT) 采用同步光纤网络/同步数字体系 (SONET/SDH) 作为 B-ISDN 的基础。由于 SONET 使用电路交换技术，并且只指定数据传输和多路复用的规范，人们需要一个新的交换技术标准。

ATM 的速度较高并且延迟较低，它的可操作版本达到了 2.5Gbps (OC-48)。ATM 在支持各种各样的技术方面也显得比较灵活，这些技术包括帧中继 (猝发)、IP 以太网、xDSL、SONET/SDH 和无线网络等。此外，它还能够保证预先定义的服务质量 (QoS)。因此，ATM 被选为 B-ISDN 的交换技术。

起初，ATM 用于 WAN，特别是作为主干。现在，它也用于 LAN 的应用中。

1. ATM 的信元结构

ATM 信元 (cell) 有固定的格式，它们的大小为 53 字节，其中前 5 字节为信元头，接下来的 48 个字节为有效载荷。

ATM 层有两种接口：用户-网络接口 (User-Network Interface, UNI) 在本地，位于用户和 ATM 网络之间，而网络-网络接口 (Network-Network Interface, NNI) 位于 ATM 交换机之间。

图 15-4 说明了 ATM UNI 信元头的结构。信元头由一个 4 位的通用流量控制 (GFC) 开始，GFC 在本地用户网络级别控制进入网络的流量。接下来是 8 位的虚拟路径标识符 (VPI) 和 16 位的虚拟通道标识符 (VCI)，它们分别用于选择特定的虚拟通路和虚拟电路。VPI (8 位) 和 VCI



(16 位) 组合起来为信元提供了一个唯一的路由指示器。打个比方, VPI 就像电话号码中的区号 (604), 而 VCI 则是区号后面的数字 (555-1212)。

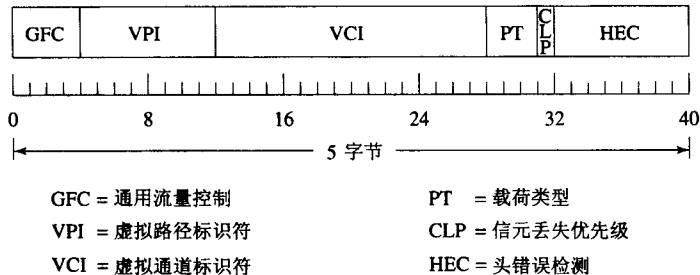


图 15-4 ATM UNI 信元头

3 位的有效载荷类型 (PT) 规定这个信元是用于用户数据还是用于管理和维护、网络拥塞等。比如, 000 表示用户数据信元类型 0, 没有拥塞; 010 表示用户数据信元类型 0, 发生拥塞。PT 可能会随网络情况而改变, 比如从 000 到 010 就表示网络变得拥塞了。

1 位的信元丢失优先级 (CLP) 允许通过设置 CLP 为 1 来表示低优先级信元。当网络拥塞的时候, 就给 ATM 交换机提供了丢弃那些信元的提示。

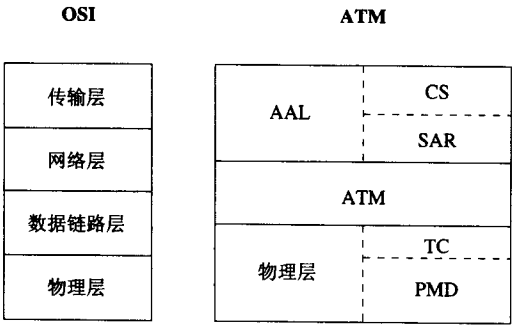
8 位的头错误检测 (HEC) 只检查信头的错误 (不检查有效负荷)。由于信头其余的部分只有 32 位, 所以这个 8 位的字段已经相对较长了。它既用于错误检查也用于错误修复[2]。

NNI 头与 UNI 头十分相似, 只不过没有 4 位的 GFC。而它的 VPI 则增加为 12 位。

2. ATM 层和子层

图 15-5 说明了 OSI 层与 ATM 适配层 (ATM Adaptation Layer, AAL) 及以下的 ATM 层和子层的之间的比较。如图所示, AAL 对应于 OSI 的传输层和一部分网络层。它由两个子层组成: 汇集子层 (CS) 及分组和重组子层 (SAR)。CS 为用户应用提供了接口 (汇集)。SAR 负责信元的分组和重组。

435  
437



- AAL = ATM 适配层
- CS = 汇集子层
- SAR = 分组和重组子层
- TC = 传输汇集层
- PMD = 物理介质相关子层

图 15-5 OSI (第四层及以下) 与 ATM 层的比较

ATM 层对应于一部分 OSI 的网络和数据链路层。它的主要功能是流量控制, 虚拟电路和通路的管理, 信元的多路复用和分离。ATM 物理层由两个子层组成: 传输汇集层 (TC) 和物理介质相

关子层 (PMD)。PMD 对应于 OSI 的物理层, 而 TC 执行信头错误检查和帧 (信元) 包装和解包装。这使得 ATM 物理层与 OSI 物理层有较大区别, 在 OSI 中分帧的工作由 OSI 数据链路层完成。

### 15.3.4 千兆和 10 千兆以太网

千兆以太网于 1998 年成为标准 (IEEE802.3z) [2]。它使用与早期以太网相同的帧格式和大小, 并向后兼容 10BASE-T 和 100BASE-T。虽然当它使用不同的光纤或者铜质介质的时候可以进一步划分为 1000BASE-LX、1000BASE-SX、1000BASE-CX 和 1000BASE-T, 但通常将它看做 1000BASE-T。1000BASE-LX 使用单模光纤 (SM 光纤) 时的最大链接距离为 5 千米, 使用多模光纤 (MM 光纤) 时的最大距离为 550 米, 使用屏蔽双绞线时则仅有 25 米。

千兆以太网对于交换机之间的连接采用了全双工模式, 对于使用中继器的共享连接则使用半双工模式。由于在半双工模式下常常会发生冲突, 所以千兆以太网像它之前的版本一样使用标准的以太网访问方法——载波侦听多路访问/冲突检测 (CSMA/CD)。千兆以太网迅速地取代了快速以太网和 FDDI, 特别是在网络主干上。它不仅仅用于 LAN, 并且在城域网中也有所应用。

10 千兆以太网于 2002 年完成。它保留了以太网的主要特征 (总线、包交换) 和同样的包格式。高达 10Gbps 的传输速率使得它只能使用光纤作为传输介质。由于它只工作在全双工模式下 (交换机和缓冲分配装置), 因此它并不需要 CSMA/CD 来进行冲突检测。

人们期望用 10 千兆以太网实现语音和数据网络的合二为一。它比 ATM 的成本更低。它的设计包含所有的 LAN、MAN 和 WAN, 并且它的传输能力等于甚至超过光纤通道、高性能并行接口 (HIPPI)、Ultra 320 或者 640 SCSI 和 ATM/SONET OC-192。它使用 SM 光纤的时候, 最大链接距离增加到 40 千米 (见表 15-5)。事实上, 技术人员对 10 千兆以太网与 SONET/SDH 的互用性也有专门的考虑, 因此以太网包能够顺利通过 SONET/SDH 链路。

表 15-5 提供了快速以太网、千兆以太网和 10 千兆以太网之间的简要比较。

表 15-5 快速以太网、千兆以太网和 10 千兆以太网的比较

	快速以太网 (100BASE-T)	千兆以太网 (1000BASE-T)	10 千兆以太网
数据率	100Mbps	1Gbps	10Gbps
传输模式	全或半双工	全或半双工	仅全双工
接入方法	CSMA/CD	CSMA/CD	N/A (无冲突)
介质	铜缆或光纤	光纤或铜缆	仅光纤
目标距离	最多 2km (光纤) 200m (铜缆)	最多 5km (SM 光纤) 550m (MM 光纤) 25m (铜缆)	最多 40km (SM 光纤) 300m (MM 光纤)
网络类型	LAN	LAN/MAN	LAN/MAN/WAN
IEEE 标准	802.3u	802.3z	802.3ae
年份	1995	1998	2002

## 15.4 接入网

接入网 (access network) 连接终端用户和主干网。它也被认为是传输各种多媒体服务的“最后一英里”, 这些服务包括 Internet 接入、电话和数字或者模拟电视服务。

除了前面讨论的 ADSL, 一些常见的接入网有:

- **混合光纤/同轴 (HFC) 电缆网络** 光纤连接主干网和邻近的光纤网络单元 (ONU), 这些单元通常为上百个家庭提供接入。这样, 所有的终端用户可以通过一条共享的同轴电缆来得到服务。

一般来说, 模拟有线电视被分配的频率范围为 50~500MHz, 划分为 NTSC TV 的 6MHz 通道和欧洲的 8MHz 通道。对于 HFC 电缆网络, 下行流被分配的频率范围为 450~750MHz, 而上行流被分配的频率范围为 5~42MHz。对于下行流, 电缆调制解调器作为调谐器来捕获 QAM 调制过的数字流。上行流则使用正交相移键控 (QPSK) [2] 调制, 因为它对于有噪声和拥塞的频谱有更好的健壮性。

HFC 的一个潜在问题是共享同轴电缆中的噪声和干扰。上行通道中的隐私和安全也是应该关心的问题。

- **光纤到社区 (FTTC)** 光纤连接主干网和社区的 ONU。每一个 ONU 通过双绞铜线或者同轴电缆连接数十个家庭。对于 FTTC, ONU 使用星型拓扑, 这样, 连接终端用户的介质并没有共享, 这相对于 HFC 是一个比较大的改善。在下行方向, 典型的传输速率为 T1~T3, 在上行方向最高为 19.44Mbps。
- **光纤到家庭 (FTTH)** 光纤直接连接主干网和一小组一小组的家庭, 提供了最高的带宽。例如, 在到达四个家庭之前, 一个 622Mbps 的下行流由 TDM 分为四个 155Mbps 的下行流。由于大多数家庭只有双绞线或同轴电缆, 所以实施 FTTH 的花费比较大。
- **地面传送** 地面广播使用 VHF 和 UHF 频谱 (大约 40~800MHz)。每个通道占用 8MHz (欧洲) 或者 6MHz (美国), 每次传输覆盖直径约为 100 千米。对于模拟视频可以使用 AM 和 FM 调制, 而对于数字视频则使用编码的正交频分多路复用 (COFDM)。这个标准称为数字视频地面广播 (DVB-T)。由于返回通道 (上行流) 并不支持地面广播, 所以常为交互式应用中的上行流使用独立的 POTS 或者 N-ISDN 链路。
- **卫星传送** 卫星广播使用千兆赫兹频谱。每个卫星可以覆盖上千千米的区域。对于数字视频, 每个卫星通道通常的传输速率为 38Mbps, 对于很多数字视频广播 (DVB) 通道来说已经足够好了。它的标准为数字视频卫星广播 (DVB-S)。与 DVB-T 类似, POTS 或者 N-ISDN 可以用于支持 DVB-S 的上行数据。

440

15.5 通用外设接口

为加以对比, 表 15-6 列出了各种用于连接 I/O 和其他设备 (硬盘、打印机、CD-ROM、指点设备 (例如鼠标)、个人数字助理 (PDA)、数码相机等) 的通用外设计口。

表 15-6 通用外设接口的速度

类 型	数据率	类 型	数据率
Serial Port	115 kbps	Ultra2 SCSI	40 MB/s
标准并行端口	115 kB/s	IEEE 1394(FireWire,i.Link)	1.5~50 MB/s
USB	1.5 MB/s	USB 2	60 MB/s
ECP/EPP 并行端口	3MB/s	Wide Ultra2 SCSI(Fast 40)	80 MB/s
IDE	3.3~16.7 MB/S	Ultra3 SCSI	80 MB/s
SCSI-1	5 MB/s	Ultra ATA 133	133 MB/s
SCSI-2(Fast SCSI, Fast narrow SCSI)	10 MB/s	Wide Ultra3 SCSI(Ultra 160 SCSI, Fast 80)	160 MB/s
Fast wide SCSI(Wide SCSI)	20 MB/s	HIPPI	100~200 MB/s
Ultra SCSI(SCSI-3, Ultra narrow SCSI)	20 MB/s	Ultra 320 SCSI	320 MB/s
EIDE	33 MB/s	光纤通道	100~400 MB/s
Wide Ultra SCSI(Fast 20)	40 MB/s	Ultra 640 SCSI	640 MB/s
USB: 通用串行总线		SCSI: 小型计算机系统接口	
ECP: 增强型高能端口		Narrow: 窄 (8 位数据)	
EPP: 增强型并行端口		Wide: 宽 (16 位数据)	
IDE: 集成磁盘电子		HIPPI: 高性能并行接口	
EIDE: 增强型 IDE			

## 15.6 进一步探索

Tanenbaum[1]和 Stallings[2]的书对计算机网络和数据通信给出了详细的论述。

441

本书网站上关于这一章的 Further Exploration 部分提供了大量介绍计算机和多媒体网络的网络资源, 包括以下链接:

- SONET FAQ 等。
- 位于 DSL 论坛的 xDSL 介绍。
- 关于 ATM 的介绍与白皮书。
- Alliance 网站上关于 10 千兆以太网的 FAQ 和白皮书。
- IEEE 802 标准。
- IETF 关于 IPv6 (IP 版本 6) 的 RFC。

## 15.7 练习

1. OSI 和 TCP/IP 参考模型之间的主要区别是什么?
2. IPv6 是一个新的 IP 协议, 与 IPv4 相比, 它有哪些优点?
3. UDP 不支持端到端的流量控制, 但是 TCP 支持。请解释这是怎么通过使用序列号实现的。给出一个例子, 其中分包后的消息使用 UDP 传输却接收到不正确的结果, 而使用 TCP 在相同的条件下 (没有通道错误) 却能正确地接收到结果。
4. 作为 FDM 的变体, WDM 用于光纤通道的多路复用。试比较 WDM 和 FDM。
5. ISDN 和 ADSL 都能为家庭用户或者小型办公室用户传输综合网络服务, 比如语音、视频等, 那么 ADSL 与 ISDN 相比有哪些优点。
6. 许多协议 (比如以太网、令牌环和 FDDI) 都常在 LAN 中使用。讨论这三种技术的功能和它们的区别。
7. 帧中继和信元中继都是包交换的变体。试比较这两种技术。
8. 交换和路由有什么区别? 路由算法是交换技术中专用的吗?
9. ATM 有多少个子层? 它们分别是什么?
10. 在 HFC 电缆网络中, 有两种调制方案用于传输下行和上行数据。为什么上行和下行要区别对待? 我们同时也应该使用不同的多路复用技术吗?

## 15.8 参考文献

- [1] A.S. Tanenbaum, *Computer Networks*, 4th ed., Upper Saddle River, NJ: Prentice Hall PTR, 2003.
- [2] W. Stallings, *Data & Computer Communications*, 6th ed., Upper Saddle River, NJ: Prentice Hall, 2000.
- [3] W. Stallings, *ISDN and Broadband ISDN, with Frame Relay and ATM*, Upper Saddle River, NJ: Prentice Hall, 1999.
- [4] K. Tolly, "Introduction to FDDI," *Data Communications*, 22(11): 81-86, 1993.
- [5] R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications & Applications*, Upper Saddle River, NJ: Prentice Hall PTR, 1995.

442

## 第 16 章 多媒体网络通信和应用

大致说来,多媒体网络通信与(传统的)计算机网络通信是类似的,因为它们都要处理数据通信。然而,多媒体网络通信比传统的计算机网络通信更有挑战性,因为多媒体数据(音频、视频等)是连续的媒体文件。它们具有以下的特点:

- **数据量大** 多媒体网络通信要求较高的数据传输率,甚至达到几十或者几百 Mbps。
- **实时性和交互性** 多媒体网络通信要求在音频和视频文件传输时保持低延时和同步性来满足“唇同步”的要求。另外,视频会议以及交互式多媒体等应用往往需要两路通信。
- **有时会出现网络数据的爆炸** 数据的传输率会发生极大的变化。比如在视频点播中,通信路径在大部分时间内是空闲的,但有时会达到很大的数据量。

### 16.1 多媒体数据传输的质量

#### 16.1.1 服务质量

多媒体数据传输的服务质量(Quality of Service, QoS)由很多的参数决定。下面就是一些比较重要的参数:

- **数据传输率** 度量传输速度的一种方法,通常以千位每秒(kbps)或者兆位每秒(Mbps)为单位。
- **延迟(最大帧/包延迟时间)** 从开始传输到接收所需的最长时间,通常以毫秒(ms)为单位。例如,在语音通信中,如果通信过程总的延迟超过 50 毫秒,就会产生比较明显的回声;如果单向延迟超过 250 毫秒,就会出现发言者的语音交叠现象,因为每个发言者在发言的时候都不知道其他发言者正在说的内容。
- **包丢失或者出错** 一种(以百分率)描述数据包传输过程中错误率的方法。数据包在因特网上的传输的过程中可能会丢失,还可能出现传输延迟和顺序颠倒的情况出现。因为通常不希望重新传输数据包,所以在实时多媒体传输过程中一个简单的差错修复方法就是在不产生严重错误的情况下重复播放出现错误前收到的最后一个包。

443

对于未经压缩的视频和音频来说,通常期望丢包率 $<10^{-2}$ (平均每 100 个数据包的传输过程中丢掉的个数)。当丢包率达到 10% 的时候,用户就会无法忍受传输的质量。对于压缩的多媒体和普通数据来说,期望的丢包率要小于 $10^{-7}$ 或 $10^{-8}$ 。一些优先级传输技术(将在 16.1.3 节中进行讨论)可以减轻丢包带来的影响和损失。

- **抖动(或延迟抖动)** 一个评价音频/视频播放过程的流畅程度的度量方法。从技术的角度来说,抖动由数据包/帧不同的延迟的变化产生。建立一个可以存放足够多的数据帧的缓冲区(抖动缓冲区),使得延迟时间更多的数据帧可以到达并存放起来,这样可以减少播放时的抖动现象。但是这样会增加整个媒体流的延时,同时这种办法在实时、交互的应用中不是很实际。图 16-1 给出了例子来说明音频/视频中的抖动。
- **同步偏离** 一种评价多媒体数据同步的度量方法,通常以毫秒为单位。对于一个高质量的唇同步传输,音频和视频的最大同步偏离应该是在 $\pm 80$  毫秒之间,同步偏离达到 $\pm 200$  毫秒的时候也可以接受。对于一个视频语音的混合文件,应保证视频领先语音的同步偏离不超过 120 毫秒,而语音领先视频的同步偏离不超过 20 毫秒。(这种视频和语音间同步偏离不

一样的情况是可能出现的，因为人们通常习惯语音落后于视频若干时间。)

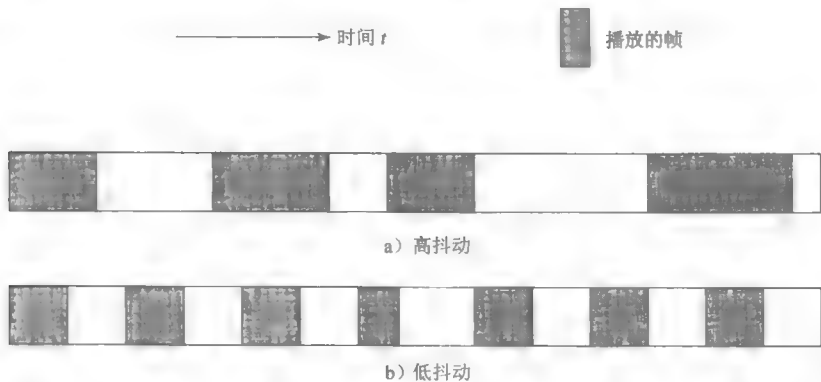


图 16-1 帧回放中的抖动

1. 多媒体服务的种类

基于以上的度量方法，多媒体应用程序可以被划分成以下几种类型：

- **实时**（也称为会晤） 双通信链路、低延迟、低抖动，可能采用具有优先级的发送方式，如视频电话和音频电话。
- **优先级数据** 双通信链路、低丢包率、低延迟，采用具有优先级的发送方式，如电子商务应用程序。
- **银级** 中度延迟、中度抖动，严格有序和同步。单通信链路的应用有视频流，双通信链路（交互性的）的应用有网络游戏等。
- **尽力型服务** 不需要实时通信，类似的应用有大文件（如电影文件）的下载和传输。
- **铜级** 对传输没有保护措施。

表 16-1 列出了一些主要的多媒体网络应用所需要占用的带宽和码率。表 16-2 列出了不同质量的数字音频和视频在传输中可以承受的最大延时和抖动。

表 16-1 网络带宽/码率的需求

应 用	速度需求
电话	16 kbps
语音会议	32 kbps
CD 质量的音乐 (QoS)	128~192 kbps 64~640 kbps
H.261	64 kbps~2 Mbps
H.263	<64 kbps
DVI 视频	1.2~1.5 Mbps
MPEG-1 视频	1.2~1.5 Mbps
MPEG-2 视频	4~60 Mbps
HDTV (压缩)	>20 Mbps
HDTV (不压缩)	>1 Gbps
MPEG-4 视频点播 (QoS)	250~750 kbps
视频会议 (QoS)	384 kbps~2 Mbps

表 16-2 数字音频和视频传输时可以承受的最大延时和抖动

应 用	平均容忍延迟 (ms)	平均延时抖动 (ms)
低端视频会议 (64kbps)	300	130
压缩语音 (16 kbps)	30	130
MPEG NTSC 视频 (1.5Mbps)	5	7
MPEG 音频 (256kbps)	7	9
HDTV 视频 (20Mbps)	0.8	1

2. 感知 QoS

尽管 QoS 通常用前面介绍的几项技术参数加以度量，但是 QoS 本身是“决定用户对该服务

的满意度的服务质量的集中表现”，上面的定义来自国际通信联盟（ITU）。也就是说，QoS 与用户的感受有很大的关系。

在实时多媒体交互的过程中，数据的平滑性要比数据的延迟性重要（也就是说，数据的抖动和波动要比稍微延长的等待时间更让人不可接受）；时序的正确性要比声音和图像的质量重要（即音频和视频的有序和同步往往是要重点考虑的因素）；人们的视觉焦点在同一时间内总是集中在一件事物上。用户通常把焦点放在屏幕的中央区域，在屏幕的内容变化之后，重新转换焦点需要一定的时间。

前面几章中我们已经讨论过人们感知的不一致性，在这一方面已经有很多关于感知的论文，它们都论述了如何使用户获得更好的感知 QoS。

### 16.1.2 IP 协议的 QoS

QoS 策略和技术可以通过向不同级别的服务提供不同的数据流或者应用来控制前面章节提到的诸如数据延迟、丢包率和抖动等关键参数。

帧中继路由协议和 ATM 提供一定级别的 QoS，但是目前几乎所有的因特网应用程序是基于 IP 建立的。IP 是一个“尽力型服务”通信技术，它不对不同的 IP 应用进行区分。这样就很难通过现有的路由方法来提供基于 IP 的 QoS。

具有充足的带宽可以提高 QoS，但是在复杂的网络世界中，我们很难在每个地方都得到足够的带宽（实际上，很多 IP 网络都在被超量使用）。特别是，充足的带宽也不能到达所有的接入连接。即使它可以到达，仅靠带宽也不能解决诸如通信链路上的数据峰值等问题。

区分服务（Differentiated Service, DiffServ）使用 DiffServ 码（IPv4 数据包中的服务类型（TOS）八位组和 IPv6 数据包中的通信量类型八位组）来对数据包分类，使得对于不同的包有不同的处理办法。尽管它也适用于端到端的网络，但它在内部局域网和企业网络内部得到了更广泛的应用，因为它很简单而且伸缩性好。区分服务（与其他 QoS 技术）一起成为事实上的 QoS 技术。读者可以查看 IETF RFC 2998 获得更多信息。

多重协议标记交换服务（Multiple Protocol Label Switching, MPLS）通过在 IP 的顶层覆盖一个协议而使得 IP 与 OSI 第二层的技术（如 ATM）结合得更紧密。它在骨干 IP 网络中引入一个 32 位的标签并把一个或者多个垫片标签放入到 IP 数据包的头中。这样就创建了一个隧道，叫做标记交换路径（Label Switched Path, LSP）。于是，骨干 IP 网络就成为面向连接的网络了。

MPLS 的优势在于，它支持流量工程（Traffic Engineering, TE）和虚拟专用网（Virtual Private Network, VPN）。流量工程主要用来控制网络流量。TE 和 VPN 都在多媒体数据的 QoS 传输中起到作用。MPLS 支持 8 种服务类型。读者可以查阅 RFC 3031 获得相关的信息。

可以将区分服务和 MPLS 结合使用，以便对每个类型的 QoS 性能和带宽提供更好的控制，同时保留 MPLS 和 DiffServ 的优点。

### 16.1.3 具有优先级的发送

当发生网络拥塞时，会使数据包丢包率和错误率提高，通过区分多媒体数据的发送的优先级可以减轻网络拥塞带来的影响。

- **媒体的优先级类型** 传输算法能够为不同的媒体数据分配不同的优先级，比如该算法设定音频数据的优先级高于视频数据的优先级，因为音频数据内容的丢失会更容易被用户察觉。
- **未压缩的音频文件的优先级** 可以把 PCM 音频位流按每  $n$  个采样分成一组，先设定它们的优先级，然后发送全部  $n$  个组中的  $k$  个组（ $k \leq n$ ），最后如果需要的话发送剩下的组给接收者，使得它们能被插入到原来的采样中。比如，如果四个数据组中有两个组丢失了，那么

有效的采样率会从 44.1kHz 降到 22.05kHz。数据的丢失被感知为采样率的变化，而不是声音的丢失。

- **JPEG 图像的优先级** 渐进式 JPEG 的不同扫描策略和分级 JPEG 图像的不同分辨率可以被赋予不同的优先级。比如，对使用 DC 系数和前几个 AC 系数的扫描方法赋予高优先级，以及对分级 JPEG 图像中的低分辨率组件赋予高优先级。
- **压缩视频文件的优先级** 视频优先级算法通过设定 I 帧的优先级最高，B 帧的优先级最低来使播放的延迟和抖动减到最低程度。在使用分层编码的视频文件中（如 MPEG-2 和 MPEG-4），通常底层的优先级要比上层的优先级高。

## 16.2 IP 上的多媒体

因为因特网具有非常大的便利性和实用性，所以人们付出了很大的努力把基于 IP 的多媒体的理论变成现实，尽管这是一个很有挑战性的任务。在本节中，我们将学习一些基于 IP 的多媒体的关键的问题、技术和协议。

### 16.2.1 IP 多播

在网络术语中，一个广播的消息被发送给域中所有节点，一个单播消息只发送到网络中的一个节点，而一个多播的消息则是发送到网络中特定的节点集<sup>①</sup>。

IP 多播使得因特网上的多播成为可能。它在邮件列表、公告板、文件组传送、音频/视频点播、音频/视频会议等应用中具有很大的作用。1988 年，Steve Deering 在他的博士论文中首先提出 IP 多播技术。1992 年的 3 月，在圣迭戈召开的 IETF 会议使用了因特网广播技术，尽管只是音频广播，但这还是 IP 多播技术发展过程中的一个重大事件。

#### 1. Mbone

因特网的 Mbone (Multicast Backbone) 是基于 IP 多播技术的[1]，它出现在 20 世纪 90 年代初期，已经在因特网的音频和视频会议中得到了应用[2, 3]。早期的应用包括用于音频会议的 vat，用于视频会议的 vic 或 nv 等。其他的应用包括共享空间中的白板 (wb) 和基于 Mbone 的会话路径管理 sdr。

因为很多路由器并不支持多播，所以 Mbone 利用支持多播的路由器的路径器 (mrouter) 来转发多播的数据包。如图 16-2 所示，路径器（所谓的“岛”）通过隧道连接。多播数据包被包装到普通的 IP 数据包内部通过隧道发送到目的地路由器，这样数据就能够在“岛”之间相互传送。

在 IPv4 中，每个 IP 地址由 32 位组成。如果前四位是 1110，就表示该消息是一个多播消息。IP 多播覆盖的 IP 地址区间从 224.0.0.0~239.255.255.255。

IP 多播中可以有匿名的成员。源主机向以上的 IP 多播地址发送消息，但它不知道谁会接收到消息。主机的软件把 IP 组地址映射到一个接收地址的列表中。这样，在硬件支持多播的时候（例如以太网和 FDDI 支持硬件多播）就发送多播消息，而在不支持多播的时候就给网络树中的邻接节点发送多个单播消息。

多播的一个潜在问题是网络中同时会存在和传输太多的数据包。幸运的是，IP 数据包具有一个生存期 (TTL) 字段，该字段可以限制数据包的生存周期。每个路由器把经过该路由器传送的数据包的 TTL 值至少减 1。当一个数据包的 TTL 值为零的时候，就将其丢弃。

上面讨论的 IP 多播方法是基于 UDP（不是 TCP）的，这样可以避免多播的接收者在收到数据包后向主机发送过多的确认消息。这样，数据包采用“尽力型服务”方式进行传输，因此数据的可靠性就受到了限制。

① IPv6 也称为泛播 (anycast)，即将消息发给任意一个指定的节点。



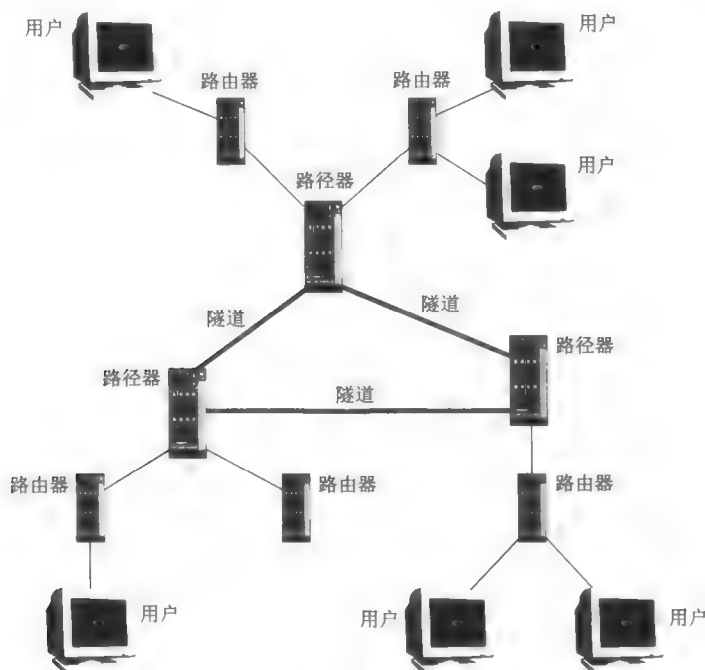


图 16-2 Mbone 中的 IP 多播隧道

## 2. 因特网组管理协议

因特网组管理协议 (Internet Group Management Protocol, IGMP) 用来帮助管理多播组。两个特殊的 IGMP 消息类型是 Query (查询) 和 Report (报告)。Query 消息由路由器发送给网络上的全部主机, 用来获取多播组的成员信息。Report 消息用来响应路由器的查询消息, 同时也可以用来加入一个多播组。

当收到一个查询信息的时候, 每个组内成员在回复信息之前要等待一段时间。如果在这段时间内, 该成员听到了其他的回复信息, 那么该成员就不会对这个查询信息作出回复。路由器定期发出多播组关系的查询, 如果收到至少一条回复的消息, 路由器便把它们设定为组内的成员; 如果在一段时间内没有收到任何回复消息, 则路由器便把它们设定为非组内成员。

IGMP 的版本 2 中要求数据延迟要低, 所以在组内成员都离开这个组的时候, 多播组的关系会被更快地清除。

## 3. 可靠多播传输

IETF PFC 2357 给出了一个评价可靠 IP 多播协议的标准。

就像 Almeroth[4]指出的那样, Mbone 维护一个平坦的虚拟拓扑结构, 而且它并不提供路径聚合 (Mbone 在峰值的时候可能有约 10 000 条路由)。这样, 它就不具有伸缩性。此外, 最初的设计还具有高分布性 (和简单性)。它假设在网络中没有专门的中央管理设备, 这会使隧道管理的效率低下, 也就是说, 网络中连接“岛”的隧道不能被恰当地分配。有时候, 在一个物理连接上创建多个隧道会造成拥塞。

Paul 等[5]提出了一个支持路径聚合和分级路由方案的可靠多播传输协议 (RMTP)。

Whetten 和 Taskale[6]对支持前向纠错 (FEC) 的可靠多播传输协议 II (RMTP II) 进行了介绍, 它主要用于多媒体数据的实时传播。

16.2.2 实时传输协议 (RTP)

448  
449

最初的因特网设计采用尽力型服务，这对于 FTP 和 E-mail 等应用来说是足够的。然而，对于实时多媒体应用，它就不是很适合了。实时传输协议用于传输实时数据（比如视频和音频流，通常应用在视频和音频会议中）。它主要用于多播，尽管它也可以应用在单播中。例如，它已经在 MBone 的视频会议 nv[3]、Netscape LiveMedia、Microsoft Netmeeting 和 Intel Videophone 中使用。

实时传输协议 (Real Time Protocol, RTP) 通常运行在 UDP 协议上，UDP 协议提供高效（但可靠性不高）的无连接的数据报服务。这里采用 UDP 而不是 TCP 的原因有两个。首先，TCP 是一个面向连接传输协议，因此在多播环境中它的容量就很难扩充。其次，TCP 协议通过重新传输错误的和丢失的数据包来达到可靠性的要求。正如我们上文中提到的，在多媒体数据传输的过程中，可靠性不是最重要的，推迟到达的重传的数据包在实时应用中是没有用处的。

既然 UDP 不能保证数据包按最初的顺序到达（不考虑多数据源的同步），实时传输协议需要创建时间戳和排序机制来保证数据的有序性。RTP 在每个数据包的头中引入了如下附加参数[7]：

- **有效载荷类型** 指出媒体数据类型和它的编码方案（比如 PCM、H.261/H.263、MPEG1、2 和 4 音频/视频等），这样接收者就会知道怎样将这些数据解码。
- **时间戳** 是 RTP 最重要的机制。时间戳用于记录数据包前八个字节被读取的时间，它是由发送者设置的。利用时间戳，接收者可以以正确的时间顺序播放视频和音频，并在多数据源（比如视频和音频）的情况下把这些流媒体同步。
- **序列号** 是对时间戳功能的一个补充。每发送一个 RTP 数据包，序列号就加 1，以保证所有的数据包能被接收者重新按顺序排列。这个要求是必须的，比如一个视频数据帧的所有数据包都具有相同的时间戳，如果只按照时间戳进行排序的话是不够的。
- **同步源 (SSRC) ID** 用于确定不同的多媒体数据源（比如音频、视频）。如果数据是从同一个源（翻译器、合成器）发出的，它们便会具有相同的同步源 ID 以保证它们的同步。
- **参与流源 (CSRC) ID** 确定参与传输的数据源，比如音频会议的所有发言者。

图 16-3 显示的是 RTP 的头格式，前 12 个八位组是固定的格式，后面的 0 或多个 32 位参与流源 ID 是可选的。

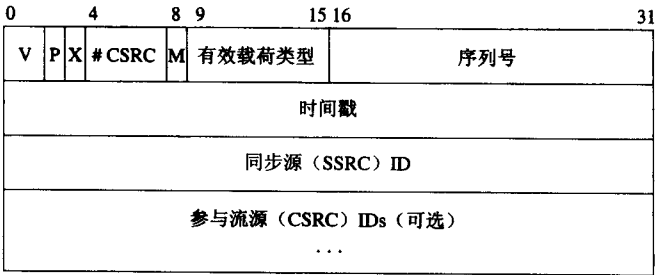


图 16-3 RTP 的包头

第 0 位和第 1 位是实时传输协议的版本号，第 2 位 (P) 声明了有效载荷，第 3 位 (X) 声明了数据头的扩展，第 4 位到第 7 位是 4 位的 CSRC 计数器，它指出在固定格式后面的 CSRC ID 的个数。

第 8 位 (M) 指出了它是音频帧的第一个数据包还是视频帧的最后一个数据包。因为音频帧在收到第一个数据包的时候即可被播放，而视频帧则须等到最后一个数到达之后才可被播放。第

9 位到第 15 位指出有效载荷的类型, 第 16 位到第 31 位是序列号, 后面跟着 32 位的时间戳和 32 位的同步源 ID。

### 16.2.3 实时控制协议 (RTCP)

实时控制协议 (Real Time Control Protocol, RTCP) 是一个与 RTP 同时存在的协议。它通过向服务器 (发送者) 提供数据传输质量的反馈来监控 QoS 的变化, 同时传递有关多方会议的参与者信息。RTCP 同时提供视频、音频同步的必要信息, 即使它们是在不同的媒体流中传送的。

RTCP 有 5 种类型的数据包:

- 1) 接收者报告 (RR) 提供传输质量的反馈 (包括最后接收到的数据包编号, 丢失的包的编号, 抖动和计算传输延迟的时间戳等)。
- 2) 发送者报告 (SR) 提供接收到的 RR 报告的信息, 包括发送的数据包的数量/字节数等。
- 3) 数据源描述 (SDS) 提供数据源的信息 (E-mail 地址、电话号码、参与者的姓名)。
- 4) 终结符表示结束。
- 5) 应用程序特定功能 (APP), 为未来的扩展特性提供空间。

RTP 和 RTCP 数据包通过不同的端口发送到相同的 IP 地址 (单播或者多播)。

### 16.2.4 资源预留协议 (RSVP)

资源预留协议 (Resource ReSerVation Protocol, RSVP) 是为了预留因特网资源而建立的协议。前面讨论的 RTP 协议并不能解决 QoS 控制的问题。开发 RSVP[8] 主要是为了得到令人满意的 QoS, 它通常是用在多播中, 尽管它也可以应用在单播中。

一个典型的被 RSVP 支持的通信模型可能由处于不同多播组的  $m$  个发送者,  $n$  个接收者组成 (如图 16-4a 所示,  $m=2$ ,  $n=3$ , 两个多播组的树分别用实线箭头和虚线箭头表明)。在广播的一个特例中,  $m=1$ ; 而在音频和视频会议中, 每个主机同时扮演着发送者和接收者的角色, 即  $m=n$ 。

RSVP 的主要挑战是多个发送者和接收者同时竞争有限的网络带宽, 接收者根据需求内容和需求 QoS 的不同而分成不同的种类, 它们还可以在任意时刻动态地加入或者退出多播组。

RSVP 中最重要的消息就是 Path 和 Resv。Path 消息由发送者创建, 同时通过多播 (或单播) 传送到目的地址。它包括有关发送者和路径 (比如前一个 RSVP 节点) 的信息, 这样接收者可以找到反向的路径, 通过它来发送资源预留消息。Resv 消息由期望得到预留资源的接收者发送。

- **RSVP 是接收者发起的** 一个接收者 (在多播扩展树的叶节点) 初始化一个资源预留请求 Resv, 该请求反向传送给数据的发送者, 但它并不是必须要传递给发送者。一个资源预留请求在路由器上可以被已经存在的、由同一会话中其他用户创建的请求合并。合并后的请求信息要求的带宽是整合在一起的所有请求中最大的带宽。这种用户发起的方案可以达到很大的规模, 同时它满足了用户的不同需要。
- **RSVP 只创建软状态** 接收者主机必须靠定期发送资源预留信息来维持这个软状态, 否则这个状态便会过期。初始化消息和后继的刷新消息间没有区别。如果接收者的资源预留要求发生了变化, 系统便会自动根据带有新参数的刷新请求信息更新状态。因此, RSVP 方案具有很高的动态性。

图 16-4 描述了一个简单的网络。该网络有 2 个发送者 (S1, S2), 三个接收者 (R1, R2, R3) 和 4 个路由器 (A, B, C, D)。图 16-4a 显示 S1 和 S2 沿着各自的路径发送 Path 信息给 R1, R2 和 R3。在图 16-4b 和 16-4c 中, R1 和 R2 分别向 S1 和 S2 发送 Resv 消息请求保留资源。在 C 到 A 的路径上, 因为 R1 和 R2 请求的是不同的数据流资源, 所以它们各自的隧道不可被合并。在图 16-4d 中, R2 和 R3 也向 S1 发送 Resv 消息请求资源。R3 的请求与前面 R1 的请求在 A 处合并,

而 R2 的请求与 R1 的请求在 C 处合并。

任何可能要求更高带宽的 QoS 变体都能通过修改保留状态的参数来进行处理。

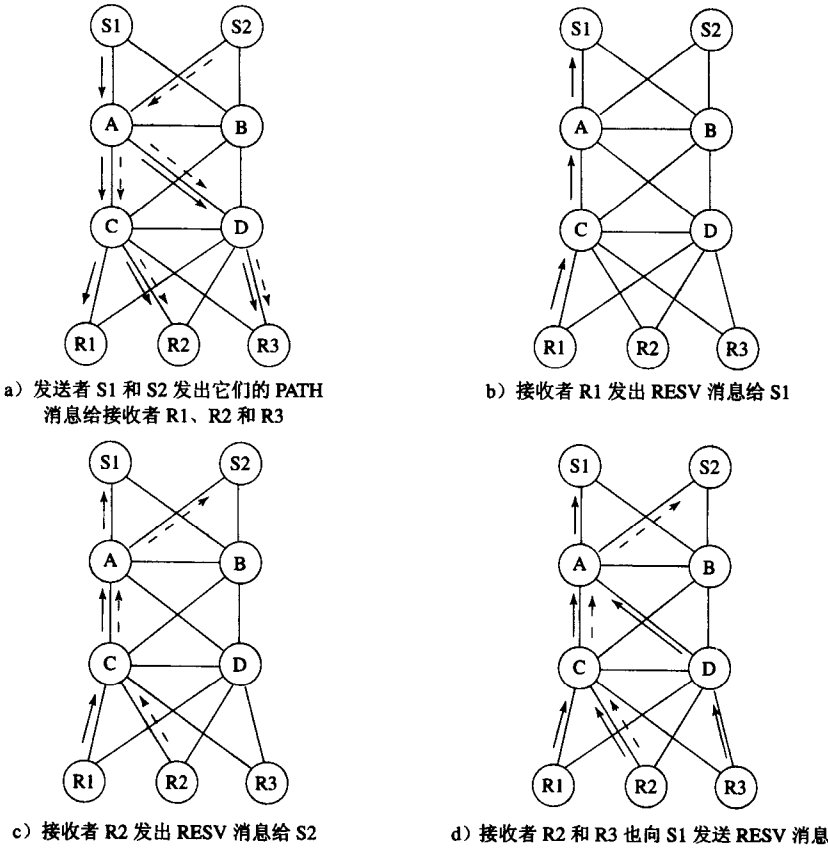


图 16-4 使用 RSVP 的网络资源预留的一个情形

16.2.5 实时流协议 (RTSP)

音频流和视频流

在早期, 多媒体数据以整个文件的形式通过网络 (通常使用低速链路) 传输, 接收者首先将它存盘, 然后在本地硬盘中打开它。近来, 越来越多的音频和视频数据通过媒体服务器以数据流——音频流和视频流的方式传输到客户端, 客户端接到数据流后立即将其解码。

通常情况下, 接收者会设置一个缓冲空间来预先取得到来的流媒体。等到缓冲区中的内容达到一定数量的时候, 缓冲区内的压缩数据会被解压然后播放。显然, 缓冲空间要足够大才能处理可能的数据抖动, 同时缓冲空间还要保证能连续、平稳的播放。另一方面, 缓冲空间太大往往会造成不必要的初始延迟, 这种情况在音频和视频会议等交互式应用中是非常不希望出现的。

实时流协议

实时流协议 (Real Time Streaming Protocol, RTSP) 用于在客户端和提供媒体资源的服务器之间进行通信, 图 16-5 指出了一种由四个 RTSP 操作组成的可能场景。

1) 请求表示描述 客户端发出一个 DESCRIBE 要求给媒体资源服务器, 希望获取服务器的表示的描述, 如媒体的类型 (音频、视频或图片等)、帧率、分辨率、编解码格式等。

450  
453

2) 建立会话 客户端发出 SETUP 信息来通知服务器发送目的地的 IP 地址、端口号、协议和 TTL (对于多播)。在服务器返回一个会话 ID 后, 该会话即被建立。

3) 请求和接收媒体 当收到 PLAY 消息后, 服务器开始使用 RTP 传送音频/视频流数据。数据流后面是 RECORD 或者 PAUSE 消息。RTSP 还支持其他的 VCR 命令, 如 FAST-FORWARD 和 REWIND 等。在会话的过程中, 客户端定期向服务器发送 RTCP 数据包来提供接收到 QoS 的反馈信息 (如 16.2.3 节中的讨论)。

4) 会话关闭 TEARDOWN 用于关闭这个会话。

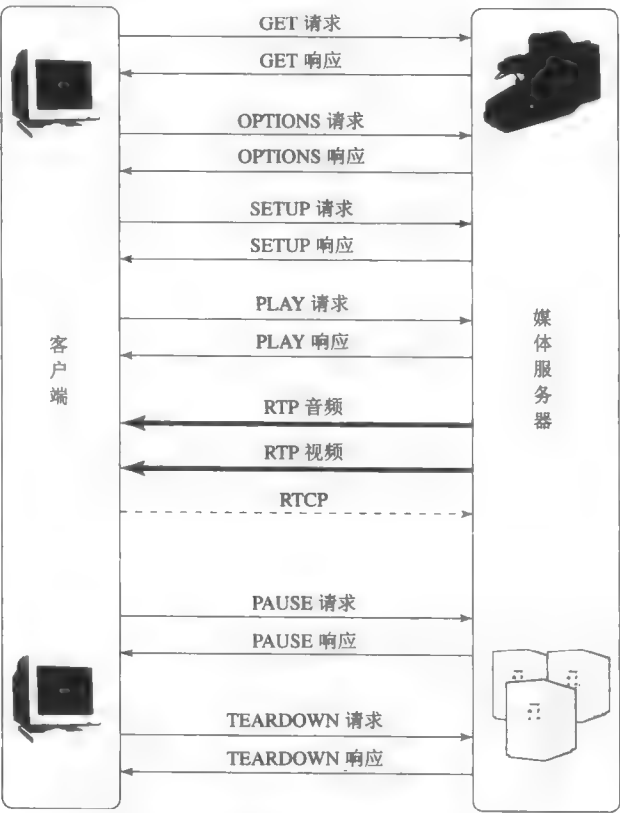


图 16-5 RTSP 操作的一个可能场景

16.2.6 因特网电话技术

公共交换电话网络 (Public Switched Telephone Network, PSTN) 依靠铜导线传输模拟声音信号。它提供了可靠而且低价的语音及传真服务。在 20 世纪 80 年代和 90 年代, 调制解调器是一种“在语音网络上传输数据”的常用方法。事实上, 它在 ADSL 和电缆调制解调器出现之前是很有优势的。

随着个人电脑和因特网更加便利和普及, 越来越多的语音和数据通信以数字形式进行 (如 ISDN)。“在数据网络上传输的语音”, 特别是 Voice Over IP (VoIP) 的出现, 吸引了大量的研究人员及用户群体的注意力。随着网络带宽不断增加, 多媒体数据压缩的质量不断提高, 因特网电话技术成为了现实。随着技术的不断发展, 因特网电话技术不仅仅局限于语音 (VoIP) ——它集成了语音、视频和数据等服务。

因特网电话技术与普通老式电话服务（POTS<sup>①</sup>）相比，主要好处如下：

- 它在集成服务（如语音邮件、视频会议、音频会议和移动电话等）方面提供更大的灵活性和扩展性。
- 它通过数据包进行交换，而不是通过数据通路进行交换。这样，网络的使用效率更高（音频通信会引起数据流量剧增，而且它使用 VBR 编码格式）。
- 随着多播或者多点通信技术的发展，多方通话技术不会比传统的两方通话技术困难很多。
- 改进的多媒体数据压缩技术可以支持不同程度的 QoS，并且可以根据网络的流量情况动态调整 QoS，这是对 POTS 中“要么全部服务，要么没有服务”的一种改进。
- 可以开发优秀的图形用户接口来显示一些可用的特性和服务，监控通话状态和通话进程等。

如图 16-6 所示，因特网电话技术的实时音频（和视频）的传输是由 16.2.2 节中描述的 RTP 所支持的（它的控制协议是 RTCP）。媒体流通过 RTSP 处理，网络资源预留是通过 RSVP 所管理的。

因特网电话技术不是简单的因特网上的流媒体服务，因为它需要一种复杂的信令协议。流媒体服务器可以通过一个统一资源标识符（URI）轻松识别；但一个因特网电话通话能否成功取决于接收者的当前位置、接收者的网络容量、可用性以及接收者是否愿意进行通话等条件的约束。

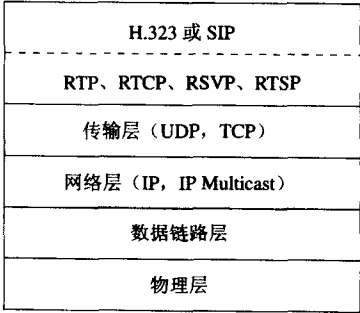


图 16-6 因特网电话技术的网络协议结构

下面对 H.323 标准和一种最常用的信令协议——会话启动协议（SIP）进行大致描述。

1. H.323

H.323[11, 12]是一个基于数据包的在无需 QoS 保障的网络上（局域网、因特网、无线网络等）进行多媒体通信服务的标准。它指定了信令协议，同时描述了终端、（视频会议的）多点控制单元，集成全球交换电话网络（General Switched Telephone Network, GSTN<sup>②</sup>）数据终端的因特网电话技术的网关。

H.323 的信令发送过程由两个部分组成：

1) 通话建立 通话的发起者首先给网关管理器（GK）发送一个登记、接纳和状态（RAS）的使用准许请求（ARQ）消息，其中包含接收方的电话号码和姓名。网关管理器可以选择批准这个请求或者给出诸如“安全性遭到破坏”或者“网络带宽不足”之类的原因来拒绝这个请求。

2) 容量交换 系统将建立一个 H.245 控制通道，它的第一步就是交换通话发起者和接收者的容量信息，比如这是一个视频、音频还是数据传输，是否经过压缩和加密等。

H.323 提供了对音频传输的强制支持和对数据和视频传输的可选择性支持。它与一系列相关的处理因特网电话通话控制与数据压缩的软件标准协同工作。下面是一些相关的标准：

① POTS 指普通老式电话服务，它不包括诸如呼叫等待、呼叫转移等新的特性。  
② GSTN 与 PSTN（公共交换电话网络）同义。

454  
~  
455

## (1) 信令与控制

- **H.225** 通话控制协议, 包括信令发出、注册、准许、数据打包和媒体流同步等。
- **H.245** 多媒体通信的控制协议。比如, 打开和关闭流媒体的通道、获取 GSTN 和因特网电话之间的网关。
- **H.235** H.323 和基于 H.245 的多媒体终端的安全与加密。

456

## (2) 音频编解码器

- **G.711** 48、56 或 64kbps 通道上的 3.1kHz 音频编解码器。G.711 描述普通电话的脉冲编码调制。
- **G.722** 48、56 或 64kbps 通道上的 7kHz 音频编解码器。
- **G.723.1** 5.3 或 6.3kbps 通道上的 3.1kHz 音频编解码器。(VoIP 论坛采用 G.723.1 作为 VoIP 的编解码器。)
- **G.728** 16kbps 通道上的 3.1kHz 音频编解码器。
- **G.729、G.729a** 8kbps 通道上的 3.1kHz 音频编解码器。(帧中继论坛采用 G.729 作为帧中继语音的编解码器。)

## (3) 视频编解码器

- **H.261** 是  $p \times 64$  kbps ( $p \geq 1$ ) 的视频编解码器。
- **H.263** 在 GSTN 上的低码率(低于 64kbps)视频的编解码器。

## (4) 其他相关的标准

- **H.320** 在 ISDN 网络上的原始视频会议标准。
- **H.324** H.320 的扩展, 用于 GSTN 网络上的视频会议的标准。
- **T.120** 实时数据和会议控制。

## 2. 会话启动协议——一个信令的协议

会话启动协议(Session Initiation Protocol, SIP) [10]是一个在因特网电话中负责建立和终止会话的应用层控制协议。这些会话不仅限于 VoIP 通信——它们还包括多媒体会议和多媒体分布。

同 HTTP 类似, SIP 是一个基于文本的协议, 这一点与 H.323 不同。SIP 同时还是一个客户端-服务器协议。通话的发起者(客户端)初始化一个请求, 服务器对它进行处理并响应。一共有三种类型的服务器。代理服务器和重定向服务器用于转发通话请求。它们之间的区别是, 代理服务器将请求转发给下一跳服务器, 而重定向服务器向客户端返回下一跳服务器的地址, 这样可以将通话请求分段重定向到目的地。

第三种类型的服务器是位置服务器, 它用来找到当前用户的位置。位置服务器通常与重定向服务器或代理服务器进行通信。这些通信可以利用 finger 协议、rwhois 协议、轻量目录访问协议(Lightweight Directory Access Protocol, LDAP), 或者其他基于多播的协议来确定用户的地址。

457

SIP 可以通过 E-mail、新闻组、网页或目录、会话通告协议(SAP, 一种多播协议)来广播它的会话。

客户端可以调用的方法(命令)有:

- **INVITE** 邀请通话接收者参与到通话中。
- **ACK** 答复这个邀请。
- **OPTIONS** 不通过建立通话来获取有关媒体属性的信息。
- **CANCEL** 终结通话邀请。
- **BYE** 终结通话。
- **REGISTER** 发送用户的位置信息给注册机(一个 SIP 服务器)。

图 16-7 给出在通话发起者初始化一个 SIP 会话时的一个可能的步骤：

**第 1 步：**通话发起者向本地代理服务器 P1 发送一个 INVITE john@home.ca 消息。

**第 2 步：**代理服务器使用 DNS 来定位 john@home.ca 的地址并返回给服务器，同时向该地址发送一个请求。

**第 3、4 步：**john@home.ca 并没有登录到服务器。重定向服务器向邻近的位置服务器发出请求，并得到 John 的当前地址 john@work.ca。

**第 5 步：**因为当前服务器是一个重定向服务器，它把 john@work.ca 的地址返回给代理服务器 P1。

**第 6 步：**通过下一个代理服务器 P2 尝试连接 john@work.ca。

**第 7、8 步：**P2 与它的位置服务器协商并获取了 John 的本地地址——john\_doe@my.work.ca。

**第 9、10 步：**连接下一个代理服务器 P3，P3 将邀请信息发送到客户端（通话接收者）。

**第 11 到第 14 步：**John 在他的本地位置同意了通话请求，并将该确认消息返回给通话发起者。

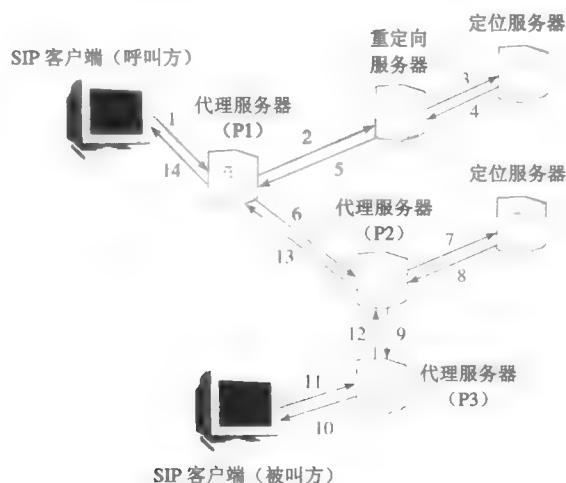


图 16-7 SIP 会话发起时的一个可能情形

SIP 也可以使用会话描述协议 (SDP) 来收集通话接收者处的媒体属性的信息。

### 3. 会话描述协议

顾名思义，会话描述协议 (Session Description Protocol, SDP) 描述多媒体的会话。同 SIP 类似，SDP 描述是文本格式的。它包括媒体流（音频、视频、白板会话等）的数量和类型，每个流的目的地址（单播或者多播），发送和接收使用的端口号以及媒体的格式（有效载荷类型）。在初始化一个通话的时候，通话发起人将 SDP 信息放在 INVITE 消息中。被叫方响应这个 SDP 信息，有时候还根据它的媒体属性修改 SDP 信息的内容。

## 16.3 ATM 网上的多媒体

### 16.3.1 ATM 网上的视频码率

ATM 论坛支持多种类型的视频码率：

- **恒定码率 (Constant Bit Rate, CBR)** 例如，在未经压缩的视频或者常数码率编码的视频中采用。上文提到过，如果给 CBR 分配的码率太低，就会出现视频内容的丢失和扭曲。



- **可变码率 (Variable Bit Rate, VBR)** 压缩视频中最常用的视频码率。可以被进一步分成适合压缩视频的实时可变码率 (rt-VBR) 和适合特定 QoS 的非实时可变码率 (nrt-VBR)。
- **可用码率 (Available Bit Rate, ABR)** 在基于 IP 的服务中, 数据传输会因为拥塞而被取消和缓冲。有时可以指定信元丢失率和最小单元数据率。
- **未定码率 (Unspecified Bit Rate, UBR)** 不提供对任何质量参数的保证。

458  
459

### 16.3.2 ATM 适配层

ATM 适配层 (ATM Adaptation Layer, AAL) 支持不同格式的用户数据和 ATM 数据流之间的相互转换。下面列出了 5 种类型的 AAL 协议:

- **AAL 类型 1** 支持实时、恒定码率 (CBR) 的面向连接的数据流。
- **AAL 类型 2** 原本用来支持可变码率的压缩视频和音频。然而, 该协议从来没有被具体化, 目前也没有被使用。
- **AAL 类型 3 和 AAL 类型 4** 它们的功能类似, 因此可以被合并成一个类型: AAL 类型 3/4。它支持面向连接或无连接的 (非实时) 数据服务的可变码率 (VBR)。
- **AAL 类型 5** 它是为多媒体数据传输而引入的新协议。它承诺支持所有类型的数据和视频服务 (包括从 CBR 到 UBR, 从 rt-VBR 到 nrt-VBR) 的支持。它假设 AAL 之上的层次是面向连接的, 而在 AAL 下面的 ATM 层是低错误率的。

如图 16-8 所示, 在汇集子层 (Convergence Sublayer, CS) 和分组和重组子层 (Segmentation And Reassembly, SAR) 中, 报头和报尾被添加到原始用户数据中。它们最终形成一个含有 5 字节 ATM 头的 53 字节 ATM 信元。

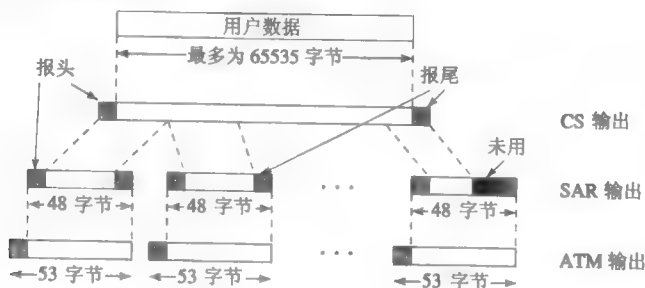


图 16-8 CS 和 SAR 子层的报头和报尾

存在五种不同类型的 AAL 很大程度上是由历史原因造成的。特别是, 除了类型 5 之外的所有 AAL 类型是由电信业开发的, 它们基本上不适合交互式多媒体的应用和服务[13]。表 16-3 给出了三种 AAL 类型之间的比较, 下面是 AAL3/4 与 AAL5 间的比较:

表 16-3 AAL 类型的比较

	AAL 1	AAL 3/4	AAL 5
CS 报头/报尾载荷	0 字节	8 字节	8 位
SAR 报头/报尾载荷	1 或 2 字节	4 字节	0 位
SAR 载荷	47 或 46 字节	44 字节	48 位
CS 校验和	无	无	4 位
SAR 校验和	无	10 位	无

- AAL 3/4 中为每个 SAR 信元指派 4 个字节的头，而 AAL 5 在这一子层上则没有头。因为有众多的 SAR 信元，省略信元的头在 AAL5 中是很大的优化，当然这只能是在现代相对无错误的光纤技术前提下才能达到的。
- 作为 SAR 报尾的一部分，AAL3/4 含有一个验证错误的校验和字段。除去 SAR 头之外，该校验和只有 10 位长，这么短的校验域是不够的。AAL5 将校验和工作放在 CS 子层上并给校验和分配 4 个字节的长度。同样的，它是基于位传输几乎不出现错误的假设建立的。然而，在 AAL5 进行错误检查的时候，它可以从这么长的校验和字段中获取足够的信息。

现在，AAL 5 已经取代了 AAL 3/4。ATM 论坛提出除了在 CBR 服务中使用 AAL 1 外，其他的服务都可使用 AAL 5。关于 AAL 的详细介绍，读者可以参考 Tanenbaum[13]和 Stallings[14]。

表 16-4 总结了使用 ATM 和不使用 ATM 的情况下视频的传输。

表 16-4 对数字视频传输的支持

视频需求	ATM 支持	非 ATM 支持
带宽	可扩展到几 Gbps	最多 100Mbps
延迟和抖动	有 QoS 支持	RSVP
CBR 或 VBR	AAL 1、2、5、LAN 竞争电路竞争等	ISDN 和 ADSL
多播	多播交换，或永久虚电路	IP 多播或独立多播协议（PIM）

16.3.3 MPEG-2 会聚到 ATM

ATM 论坛认为，MPEG-2 应该通过 AAL5 协议来传输。正如我们在 11.3.3 节中提到的那样，在默认情况下，传输数据流（TS）中的每两个 MPEG-2 数据包（每个包 188 字节）将会被映射到一个 AAL5 服务数据单元中[15]。

在建立一个虚拟的通道连接的时候，必须确定下面的 QoS 参数：

- 最大信元传输延迟（延时）
- 最大信元延迟抖动
- 信元丢失率（CLR）
- 信元错误率（CER）
- 信元块严重出错率（SECBR）

一个视听特定服务收敛子层（Audio-Visual Service-Specific Convergence Sublayer，AVSSCS）也被提出，用来在 AAL5 上使用 ABR 服务进行视频传输。

16.3.4 ATM 上的多播

与在 UDP 上作为“尽力型”服务提供的 IP 多播相比，ATM 上的多播网络有了更多的挑战 [16，17]：

- ATM 是面向连接的，因此，ATM 上的多播必须建立全部的多点连接。
- ATM 上的 QoS 必须在连接建立的时候进行协商，并且被全部的交换机所知晓。
- ATM 很难支持多点对点或多点对多点的连接，因为 AAL 5 并不保存多路复用器的编号或者顺序号。如果在接收的时候多个发送端的信元有一定的重叠，则数据就不能在接收端重新正确组装。

可扩展的有效 ATM 多播（Scalable and Efficient ATM Multicast，SEAM）和共享多对多 ATM 预留（Shared Many-to-Many ATM Reservation，SMART）是两种在 ATM 上进行多播的方法。前者使用了唯一的标识符，而后者则使用了令牌方案来避免信元重叠造成的混淆。

460  
~  
461

## 16.4 MPEG-4 的传输

MPEG-4 的设计是由万维网上的多媒体应用而激发的。特别是,多媒体(文本、图像、视频、音频等)对象和场景描述(视频对象的时间空间描述)由服务器发出,并由客户端解释并重新组装,这样可以极大地减少多媒体数据在万维网上的传输。本节主要描述多媒体传送集成框架(Delivery Multimedia Integration Framework, DMIF)和 IP 上的 MPEG-4 等问题。

### 16.4.1 MPEG-4 中的 DMIF

DMIF 是多媒体应用程序和多媒体数据传输之间的接口。它支持远程交互式网络访问(IP、ATM、PSTN、ISDN 或者手机)、广播媒体(卫星或者电缆)和磁盘上的本地媒体。

因为接口对应用程序来说是透明的,所以只要建立正确的 DMIF,单一个程序就可以在不同的传输层上运行。

图 16-9 显示了三种通信媒体的传输集成。如图所示,本地应用与统一的 DMIF 应用程序接口(DMIF Application Interface, DAI)进行交互,将应用的请求翻译成特定的协议信息并利用三种媒体类型的一种进行传输。

462

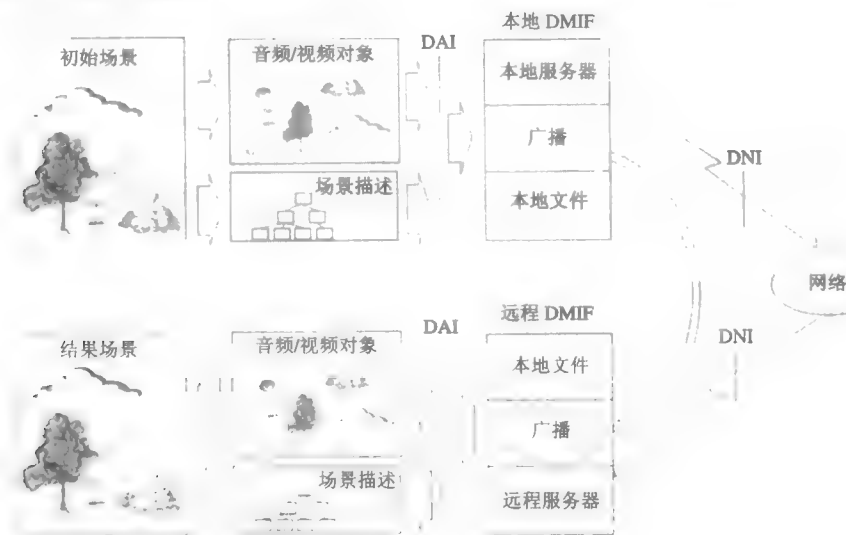


图 16-9 DMIF——多媒体内容传输集成框架

当在网络中进行传输的时候,DMIF 并不知道应用程序的存在。实际上,为了管理特定网络的信令消息,我们还需要额外的 DMIF 网络接口(DMIF Network Interface, DNI)。

当传输多媒体数据的时候,DMIF 与 FTP 很类似。首先,使用 SETUP 与远程网络站点建立一个会话。第二步,选择适当的流,并同时发送 STREAM 请求信息给 DMIF,返回一个指向发生流传输的独立连接的指针。第三步,建立新连接,数据以流的形式传输。

在广播和本地存储的情况下中,应用程序要知道数据是怎样被保存和传输的。这样,它成为了 DMIF 实现的一部分。

DMIF 内置在 QoS 监控功能中,它支持持续监控、特定的 QoS 查询以及 QoS 违反通知。

### 16.4.2 IP 上的 MPEG-4

IP 网络上的 MPEG-4 的规范是由 MPEG 和 IETF 共同开发的,它是 MPEG-4(ISO/IEC14496-8)第 8 部分的框架和 IETF 中的一个提供信息的 RFC。

MPEG-4 会话可以被 RTP、RTSP 和 HTTP 等基于 IP 的协议所传输。关于 RTP 有效载荷格式的细节在 IETF RFC 3016 中有详细的描述。简而言之,一般 RTP 有效载荷格式(generic RTP payload format)定义了从逻辑 MPEG-4 SL 数据包到 RTP 数据包的映射,而 FlexMux 有效载荷格式(FlexMux payload format)则定义了 FlexMux 数据流与 RTP 数据包的映射。

463

## 16.5 媒体点播

媒体点播(Media-on-demand, MOD)涉及许多基本的多媒体网络通信问题。在本节中,我们将简要介绍交互式电视、视频点播的广播方案,以及缓冲区管理的一些内容。

### 16.5.1 交互式电视和机顶盒

交互式电视(Interactive TV, ITV)是基于家庭电视的多媒体系统。它支持不断增长的多媒体交互方式,比如:

- 电视(普通电视、订阅电视、付费电视)
- 视频点播(VOD)
- 信息服务(新闻、天气、杂志、体坛风云等)
- 交互式娱乐(网络游戏等)
- 电子商务(在线购物、股票交易)
- 访问数字图书馆和教学资源

数字视频广播(Digital Video Broadcasting, DVB)的一个新的发展方向是多媒体家庭平台(Multimedia Home Platform, DVB-MHP),它支持上述的所有功能,同时还支持电视上的电子节目指南(EPG)。

ITV 与常规的有线电视的主要区别在于,ITV 提供了与用户的互动,所以它需要两路的通信——上行数据流(用户到内容提供商)和下行数据流(内容提供商到用户)。另外,ITV 具有更丰富的信息和多媒体内容。

为了实现 ITV 的上述功能,用户还需要机顶盒(Set-Top Box, STB)。如图 16-10 所示,机顶盒主要由如下组件构成:

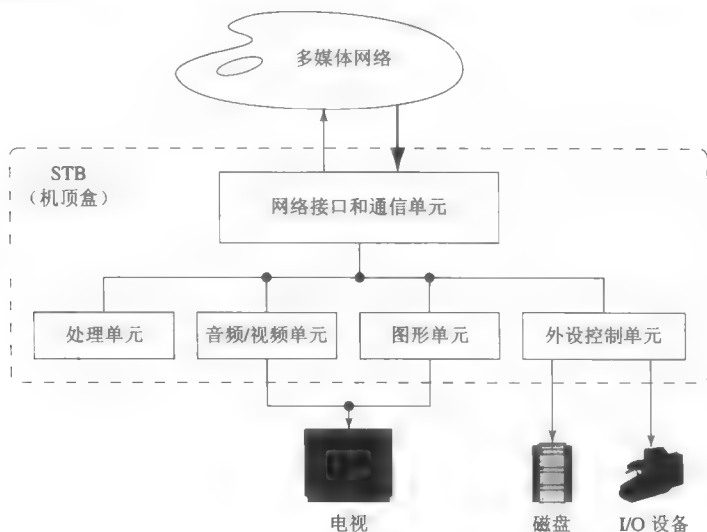


图 16-10 机顶盒的一般结构

- **网络接口和通信单元**：包括调制器、解调器（用来从模拟频道信号中获得数字流）、安全设备，以及一个用于万维网基本协商和访问数字图书馆而设置的专用通信通道，它同时用来提供其他服务和进行维护。
- **处理单元**：包括 CPU、存储器和为 STB 设计的特殊用途的操作系统。
- **视频/音频单元**：包括音频和视频（MPEG-2 和 4）解码器、数字信号处理器（DSP）、缓冲区和 D/A 转换器。
- **图形处理单元**：支持实时 3D 动画和游戏的图形处理。
- **外设控制单元**：磁盘、音频、视频 I/O 设备（如数字摄像机）、CD/DVD 读写设备等的控制器。

15.4 节中我们介绍了不同的接入网，同时，我们还对它们在 ITV 服务中传输多媒体数据的效率和安全性进行了比较。

### 16.5.2 视频点播的广播方案

在所有可能的媒体点播服务中，最受欢迎的大概就是电影的订阅服务：在高速的网络中，顾客可以指定他们想要看的电影和电影的播放时间。对于类似服务的统计表明，绝大多数顾客想看的电影集中在少数（10~20 部）流行的电影中（比如，新上映的电影和每季最受欢迎的 10 部电影）。这使得对这些电影进行多播和广播成为可能，因为许多点播电影的客户端可以按照他们的需求被放入到相应的多播组中。

评价此类 MOD 服务的重要的品质参数是等待时间（延迟）。我们定义接入时间为从客户点播电影到电影开始播放所需等待时间的上界。

因为我们已经知道光纤网络可以提供极高的带宽，所以我们可以确信，如果用户接入某个高速的网络中，整个电影可以在相对较短的时间内从服务器传给客户端。这种方式的一个问题是用户要在客户端准备足够的缓冲空间。

#### 1. 交错广播

为简单起见，我们假设所有的电影都是以固定码率（CBR）来进行编码的，所有电影具有相同的长度  $L$ （以时间单位来进行度量），并且从头到尾连续播放没有停顿。用可以达到的带宽  $W$  除以媒体的播放速率  $b$  得到带宽比  $B$ 。服务器的带宽通常被分成  $K$  个逻辑通道（ $K \geq 1$ ）。

假设服务器最多向外播放  $M$  部电影，所有电影可以在不同通道上循环播放。这样就引入了一个交错广播的问题。图 16-11 显示了一个交错广播的例子，在本例中， $M=8$ ， $K=6$ 。

如果服务器的带宽被均匀划分成  $K$  个逻辑频道，那么每部电影的接入时间就为  $\delta = \frac{M \cdot L}{B}$ （注意：接入时间实际上与  $K$  的值无关）。换句话说，接入时间会随着网络带宽的增加而线性减少。

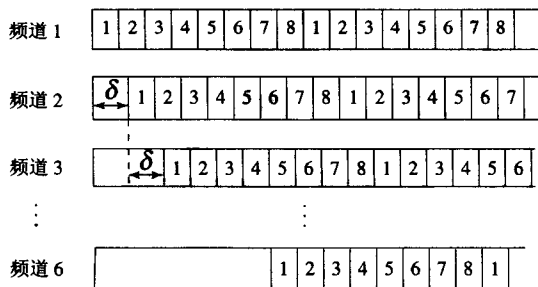


图 16-11 交错广播，此例中有 8 部电影（ $M=8$ ），6 个通道（ $K=6$ ）

## 2. 金字塔广播

Viswanathan 和 Imielinski[18]提出了金字塔广播的假设。在这个假设中,将电影文件划分成长度依次增加的段。就是说,  $L_{i+1} = \alpha \cdot L_i$ , 其中  $L_i$  是第  $i$  段  $S_i$  的大小(长度),  $\alpha > 1$ 。 $S_i$  会在通道  $i$  上周期性地广播。换句话说,交错广播是在  $K$  个通道上以一个电影位为单位播放,而在金字塔广播中是以电影的段为单位,电影不交错。每个通道具有相同的带宽,所以对于长度较大的段,传输的频率就会比较小。

因为我们已经假设可用的带宽要比电影的播放速率  $b$  大(即  $B \gg 1$ ),所以根据他们的假设,金字塔广播中用户可以在播放一个较小的段  $S_i$  中同时接收一个更大的段  $S_{i+1}$ 。

为了保证播放能够连续进行(没有停顿),必要的条件是:

$$\text{playback\_time}(S_i) \geq \text{access\_time}(S_{i+1}) \quad (16.1)$$

其中,  $\text{playback\_time}(S_i) = L_i$ 。假设分配给每个通道的带宽是  $B/K \cdot b$ , 接入时间  $\text{access\_time}(S_{i+1}) = \frac{L_{i+1} \cdot M}{B/K} = \frac{\alpha \cdot L_i \cdot M}{B/K}$ , 我们得到:

$$L_i \geq \frac{\alpha \cdot L_i \cdot M}{B/K} \quad (16.2)$$

从而有:

$$\alpha \leq \frac{B}{M \cdot K} \quad (16.3)$$

$S_1$  的大小决定金字塔广播的接入时间。默认情况下,我们设定  $\alpha = \frac{B}{M \cdot K}$  来得到最小的接入时间。随着带宽  $B$  的增加,接入时间以指数幅度减小,因为  $\alpha$  是线性增加的。

上述方案的一个缺点是客户端需要大量的存储空间,因为电影的最后两个段的总和大约占整个电影大小的 75%~80%。为了取代按几何级数增长的序列,我们使用摩天大楼式广播[19]。在这种方案中,段长的序列为  $\{1, 2, 2, 5, 5, 12, 12, 25, 25, 52, 52, \dots\}$ , 以此减轻对大缓冲区的要求。

图 16-12 显示的是一个含有 7 个段的摩天大楼式广播的例子。如图所示,两个客户端分别在不同的时间间隔(1, 2)和(16, 17)提交了点播的请求,它们有不同的传输时间安排。在任何给定时刻,客户端最多能接收两个段。

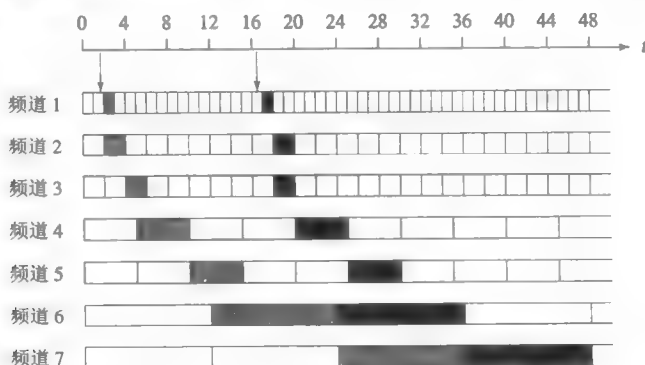


图 16-12 有 7 个段的摩天大楼式广播

Hu[20]在 2001 年描述了贪婪等带宽广播 (GEBB) 算法。段的大小和它们相应通道的带宽是

以获得广播特定视频所需的最小总服务器带宽为目的而确定的。和上面提到的基于金字塔原理的广播方案不同, GEBB 采用了一种比较“贪婪”的方式。当接入到视频广播中后, 客户端立刻开始尽可能多地接收服务器的数据, 客户端只在开始播放某个电影段的时候才停止接收该段的内容。图 16-13 描述了 GEBB 的一个例子, 在该图中, 所有的通道带宽是相等的。

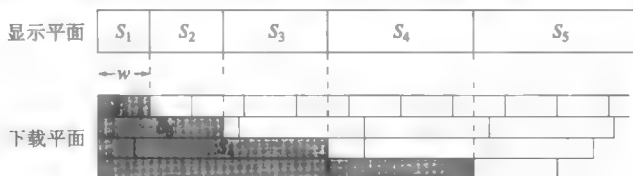


图 16-13 GEBB 示例, 阴影区域表示客户端接收和播放的数据

服务器带宽优化问题可以被形式化的陈述为:

$$\begin{aligned} & \text{minimize } \sum_{i=1}^K B_i \\ & \text{subject to } B_i = \frac{S_i}{w + \sum_{j=1}^{i-1} S_j} \quad (i=1, 2, \dots, K) \end{aligned} \quad (16.4)$$

其中,  $w$  为等待时间,  $B_i$  是第  $i$  个通道的带宽。式 (16.4) 所示的条件可以保证段  $S_i$  可以在段  $S_{i-1}$  播放终止的时候能够被接收完成。这样, 所有的段就能够连续播放。

上述的非线性优化问题可以用拉格朗日乘子规则来解决。得到的结果是, 所需的总带宽在所有的通道带宽都相等的情况下最小。每个通道的广播带宽为:

$$B_i = B_j = B^* \quad (1 \leq i, j \leq K) \quad (16.5)$$

$$B^* = \left( \frac{S}{w} + 1 \right)^{\frac{1}{K}} - 1 \quad (16.6)$$

$$S_i = \left[ \left( \frac{S}{w} + 1 \right)^{\frac{1}{K}} - 1 \right] \left( \frac{S}{w} + 1 \right)^{\frac{i-1}{K}} \quad (16.7)$$

段的大小成几何级数序列分布。 $\left( \frac{S}{w} + 1 \right)^{\frac{1}{K}}$  称为视频段的黄金系数。

### 3. 谐波广播

Juhn 和 Tseng[21]在 1997 年提出了谐波广播的概念, 它采用了一种新的广播策略。在该策略中, 每个段的大小都相同, 但是第  $i$  个通道的带宽是  $B_i = b/i$ , 其中  $b$  是电影的播放速率。换句话说, 通道带宽依次为:  $b, b/2, b/3, \dots, b/K$ 。用于电影传输的总带宽为:

$$B = \sum_{i=1}^K \frac{b}{i} = H_K \cdot b \quad (16.8)$$

$K$  是电影文件的段总数,  $H_K = \sum_{i=1}^K \frac{1}{i}$  是  $K$  的谐波系数。

图 16-14 是一个谐波广播的例子。在提出播放请求后, 客户端开始下载并从通道 1 播放电影文件的第一段  $S_1$ 。与此同时, 客户端还从相应的通道下载电影的其他各个段。

以  $S_2$  为例, 它包括两个部分  $S_{21}$  和  $S_{22}$ 。因为  $B_2$  的带宽只有  $b/2$ , 所以在  $S_1$  的播放过程中, 只有  $S_{21}$  能被下载 (预下载)。客户端可以在  $S_2$  播放的时间内下载  $S_2$  的另一部分  $S_{22}$ , 播放结束的时候,  $S_2$  刚好被全部下载完成。类似地, 在  $S_2$  播放结束的时刻, 客户端已经提前下载了  $S_3$  的  $2/3$ ,  $S_3$  剩下的  $1/3$  可以在  $S_3$  播放完成的时刻被下载完毕, 因为  $S_3$  占用的通道 3 的带宽是  $b/3$ 。

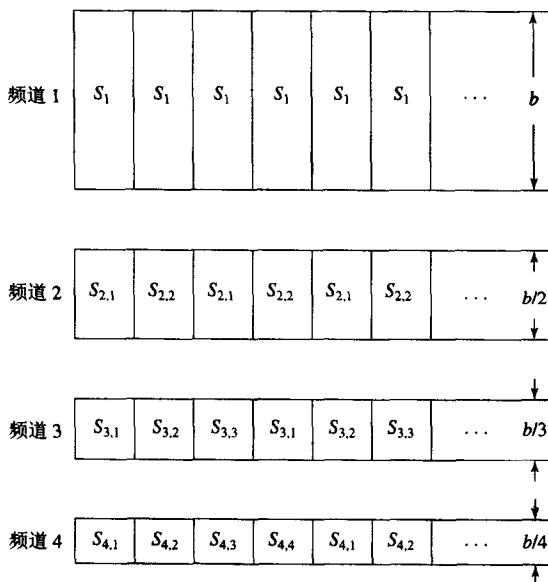


图 16-14 谐波广播

谐波广播的好处是谐波系数随着  $K$  的变大而增长缓慢。比如, 在  $K=30$  的时候,  $H_k \approx 4$ 。如果电影长度为 120 分钟, 每段的长度为 4 分钟 ( $120/30$ )。这样, 谐波广播的接入时间通常要比金字塔广播的接入时间短, 而且占用的带宽也比较小 (在本例中为  $4b$ )。Juhn 和 Steng[21]指出, 客户端缓冲区的容量上界大概是整个电影大小的 37%, 与原始的金字塔广播方案相比, 这还是很不错的。

469

然而, 上述的谐波广播方案有时也会出现问题。比如, 如果客户端从图 16-14 所示的第二个  $S_1$  开始下载, 当它完成的时候, 只有  $S_2$  的后一半  $S_{22}$  被预下载。客户端便不能够从通道 2 同时播放和下载  $S_{21}$ , 因为通道 2 的带宽只有播放速率的一半。

对于上述谐波广播的问题的一个改进是服务器请求客户端延迟播放  $S_1$  一段时间。这种延迟的谐波播放方案的缺点是它的接入时间会成倍增加。1997 年之后, 又出现了多种原始谐波广播方案变体, 如谨慎的谐波广播、准谐波广播和多重谐波广播等, 它们都解决了上述问题, 但增加了方案的复杂性。

#### 4. 宝塔广播

谐波广播方案使用大量的低带宽媒体流来广播视频, 而金字塔广播方案使用少量的高带宽媒体流广播视频。金字塔广播方案需要的总带宽要比谐波广播需要的带宽高。但是, 在谐波广播中管理数量巨大的独立媒体流常常让人很伤脑筋。

Paris、Carter 和 Long[22, 23]提出了宝塔广播和它的变体。他们提出了一个将谐波广播和金字塔广播方案的优点结合起来的广播策略。

图 16-15 解释了宝塔广播。它把每一个视频文件分成了  $n$  个固定大小的段, 每一个时间段的大小  $T=L/n$ 。服务器以视频的播放速率  $b$  为带宽广播每个电影段, 但是各段的周期不一样。所以



宝塔广播的问题在于选择合适的从段到通道的映射关系以及各个段的播放时间。

与金字塔广播和谐波广播相比, 宝塔广播对于任何给定的等待时间而言, 并不都是其带宽利用高效的。与谐波广播相比, 它需要相对较少的段来满足适当的等待时间要求, 但与金字塔广播相比则需要更多的段。

以上的所有协议是基于视频文件采用固定码率 (CBR) 来编码的假设建立的。有些协议可以用来处理可变码率编码的视频。关于这个问题的更多知识, 读者可以参考[20, 24]。

时段	1	2	3	4	5	6	7	8	9	10	11	12	13
频道 1	$S_1$	$S_1$	$S_1$	$S_1$	$S_1$	$S_1$	$S_1$	$S_1$	$S_1$	$S_1$	$S_1$	$S_1$	$S_1$
频道 2	$S_2$	$S_4$	$S_2$	$S_5$	$S_2$	$S_4$	$S_2$	$S_5$	$S_2$	$S_4$	$S_2$	$S_5$	$S_2$
频道 3	$S_3$	$S_6$	$S_8$	$S_3$	$S_7$	$S_9$	$S_3$	$S_6$	$S_8$	$S_3$	$S_7$	$S_9$	$S_3$

图 16-15 宝塔广播的前 3 个通道-段映射

### 5. 流合并

上述广播方案在希望限制用户交互时是很高效的, 即一旦发出请求, 客户端并不做其他动作而是让服务器选择调度策略并等待整个电影播放完毕。

流合并对于动态用户交互来说具有更好的适应性, 动态用户交互是通过动态合并多播会话来达到的。它仍然假设客户端接收带宽要比视频的播放速率高。实际上, 通常假设接收带宽至少是播放速率的两倍以上, 这样客户端可以同时接收两个视频流。470

一旦收到客户端的视频点播请求, 服务器就开始向客户端发送媒体流。同时, 客户端还能访问同一视频的另一个流, 这个媒体流是之前由其他的客户端初始化的。在某一个时刻, 第一个媒体流就失去作用了, 因为客户端已经从第二条媒体流中获得了媒体的其他内容。此时, 第一个流会与第二个流合并 (或称第一个流加入到第二个流中)。

如图 16-6 所示, “第一个流” B 在  $t=2$  开始。实线表示媒体播放的速率, 而虚线则指出了接收流媒体的带宽, 它是播放速率的两倍。客户端从媒体流 A 接收视频, A 在  $t=0$  时被初始化。在  $t=4$  时, 媒体流 B 加入到 A 中。

流合并的技术可以分级方式应用, 即分级多播流合并 (Hierarchical Multicast Stream Merging, HMSM)。如图 16-16 所示, 媒体流 C 在  $t=4$  的时刻开始, 在  $t=6$  的时刻加入 B, 而 B 加入 A。最初的流 B 本来将在加入 A 之后 ( $t=4$ ) 失去作用。在这种情况下, 它还要保持到 C 加入 A ( $t=6$ )。

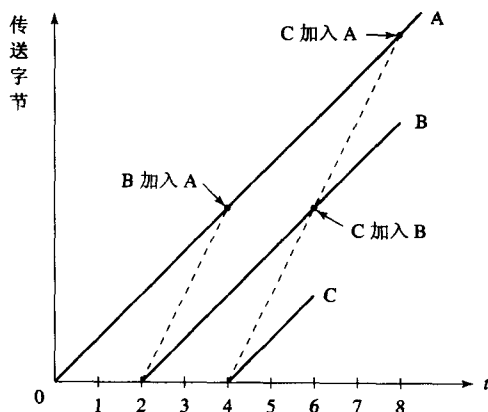


图 16-16 流合并

流合并的一个变体是“搭便车”(piggybacking), 在该算法中, 流的播放速率可以做动态的、轻微的调整, 从而使流合并(搭便车)成为可能。

### 16.5.3 缓冲区管理

连续播放的媒体通常具有期望的播放速率, 比如, 对于 NTSC 视频为 30 帧每秒, 对于 PAL 视频为 25 帧每秒。如果视频是通过网络传播的, 那么在不使用播放时的预平滑技术的条件下, 所需要的网络吞吐量必须比视频文件的峰值码率要高, 这样才能保证不间断地播放。

正如前面所讨论的, 绝大多数压缩媒体是采用可变码率方式编码的。通常情况下, 文件的活动越多(视频文件中的动作, 音频文件中的语音的变化等), 所需要的码率越高。MPEG-1 的平均码率为 1.5Mbps, MPEG-2 的平均码率大于等于 4Mbps。具有可变码率特征的媒体可能在某一点具有很高的码率, 而在另一点具有比较低的码率。媒体的峰值码率可能会远高于媒体的平均码率, 因而不能被网络的可用带宽所支持。

尽管在当今并不常用, 固定码率编码也是一个可选的方案, 即为了保证固定码率, 可能会出现部分数据的失真。固定码率的编码的效率比可变码率编码效率低: 为了得到播放质量相当的已编码媒体, 固定码率编码的码率要比可变码率编码的码率(平均码率)高 15%~30%。

为了处理可变码率以及网络负载的连续性, 我们通常在服务器端和客户端引入缓冲区[9]。客户端(例如, 在客户端的机顶盒)要引入一个预取缓冲区(prefetch buffer)来使传输率更加平滑(减少峰值速率)。假设帧  $t$  的大小为  $d(t)$ , 缓冲区的大小为  $B$ , 当前接收到的数据字节大小为  $A(t)$ (在  $t$  帧播放时), 那么对于所有的  $t \in 1, 2, \dots, N$ , 应该有:

$$\sum_{i=1}^t d(i) \leq A(t) \leq \sum_{i=1}^{t-1} d(i) + B \quad (16.9)$$

当  $A(t) < \sum_{i=1}^t d(i)$  时, 网络并没有被充分使用, 这样缓冲区处于下溢(饥饿)状态, 而当

$A(t) > \sum_{i=1}^{t-1} d(i) + B$  时, 网络容量被超额使用, 缓冲区溢出。在这两种情况下, 我们都不能平稳、

连续地播放视频文件。在缓冲区饥饿的状态下, 没有视频数据可以播放; 而在缓冲区溢出的状态下, 溢出的部分会被丢弃。

图 16-17 说明了被媒体播放(消耗)速率和缓冲数据速率所限制的网络带宽情况。(传输的速率为曲线的斜率。)在任意一个时刻, 为了保证流畅地播放, 数据必须保存在缓冲区中, 同时数据传输的速率应该大于数据消耗的速率。如果网络带宽可以达到图中的第二条直线所示的水平, 那么在播放进行的某一时刻, 被消耗的数据要比接收到的数据多。这时缓冲区就会处于饥饿状态, 播放便会终止。同时, 在任何一个时间点上, 传输的数据总量不能超过已经播放的数据总量再加上缓冲区的大小。

如果网络带宽可以达到图中的第一条直线所示的水平, 同时媒体文件尽可能占用所有的带宽来传送, 不必考虑缓冲区的影响(像通常的文件下载一样), 那么如图所示, 在快到达文件的末尾处时, 接收的数据量会超过缓冲区能同时存储的数据量。缓冲区便会溢出, 同时丢弃多余的数据包。服务器将重新传输这些被丢弃的数据包, 或者选择将这些数据包直接丢弃不再重新发送。

尽管在溢出的时候, 可以用一定的时间来重新传送数据, 但这会增加占用的带宽(可能引起将来的缓冲区饥饿)。在许多种情况下, 比如广播, 通常没有类似的反向信道。

维持预下载缓冲区中的数据既不溢出也不饥饿的技术称为传输率控制方案。有两个简单的

策略, 一是充满缓冲区后以平均视频码率来进行传输, 另一个策略是在不超过可用带宽的前提下保证缓冲区处于充满状态。如果视频需要的带宽高于现有带宽, 那么传输率控制方案假设部分数据已经被预取在缓冲区中, 可用到的网络带宽可以保证在连续播放时不会出现缓冲区的饥饿现象。

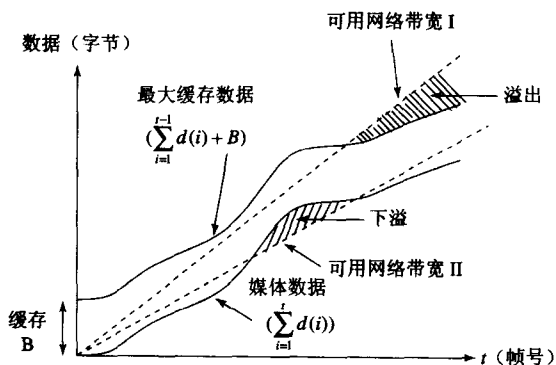


图 16-17 当媒体速率超出可用的网络带宽时, 存放在缓存中的数据能帮助顺利地播放媒体

#### 传输率的一种优化设计

介绍了服务器上存储的媒体的数据率特性后[26], 我们可以更加有效地利用预取缓冲区来实现数据的网络传输。媒体服务器可以在传输进行之前假定一个传输速率, 使得媒体可以连续地播放, 同时使保留的带宽达到最小。大部分传输计划可以使得尖峰速率达到最小, 但还存在一种特殊的策略可以使速率的可变性, 即传输率的变化达到最小。这样的传输控制计划称为最佳超前流畅设计。

使速率的可变性达到最小是很重要的, 因为这意味着该最佳速率设计使得视频分成以常数速率传输的小段的集合。根据当前传输率而不是尖峰速率确定带宽保留策略, 可以使运算处理和网络资源占用尽可能小, 同时保留带宽的变化会尽可能少。为了便于讨论, 下文提到的均为视频媒体, 尽管该技术也可扩展到传统意义上的媒体。

视频数据率可以以帧为单位分析, 但这种方法不是最佳的策略, 因为内部的帧不能被它们自己解码, 因而会产生一个解码延迟。另外, 该算法的计算成本也比较高。更实际的做法是根据播放每个 I 帧时消耗的数据量来估算视频的数据速率。假设在相同的场景中, 电影的数据速率是常数, 那么若只考虑每个场景变换后的第一帧数据量, 这种估算可能比较粗略。

像从前一样, 假设  $d(t)$  是帧  $t$  的大小 ( $t \in 1, 2, \dots, N$ ),  $N$  是视频文件的总帧数。类似的, 定义  $a(t)$  为视频服务器在第  $t$  帧播放时 (为了简单, 我们称作在时间  $t$  内) 传输的数据总量。设  $D(t)$  为消耗的总数据量,  $A(t)$  为在时间  $t$  收到的总数据量。于是有:

$$D(t) = \sum_{i=1}^t d(i) \quad (16.10)$$

$$A(t) = \sum_{i=1}^t a(i) \quad (16.11)$$

设缓冲区的大小为  $B$ 。在任意的时间  $t$ , 在没有造成缓冲区溢出的情况下, 接收的最大总数据量为  $W(t) = D(t-1) + B$ 。现在我们可以很容易地描述出使得缓冲区既不饥饿也不溢出的服务器传输速率需要满足的条件:

$$D(t) \leq A(t) \leq W(t) \quad (16.12)$$

为了在视频的持续时间内避免出现缓冲区的饥饿和溢出, 式(16.12)应该对所有的  $t \in 1, 2, \dots, N$  成立。定义  $S$  为服务器传输计划, 即  $S = a(1), a(2), \dots, a(N)$ , 如果  $S$  对式(16.12)成立, 那么  $S$  就称为可行的传输计划。图 16-18 描述了包围的曲线  $D(t)$  和  $W(t)$ , 同时图中还显示, 一个常数(或平均)码率的传输计划对于该视频不是可行的, 因为只是采用平均码率会导致缓冲器饥饿。

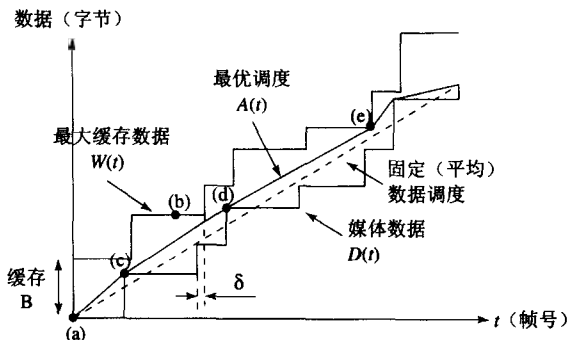


图 16-18 一个特定的视频的优化的平滑策略和缓冲区大小。在此例中, 不适合以固定的数据率传输

如果在传输之前已经知道了所有帧的大小  $d(t)$ , 那么服务器可以提前生成一个优化的传输计划, 该计划可行并且使传输尖峰速率尽可能小[26]。另外, 该计划使得计划中的变化达到最小, 使传输尽可能地平稳。

我们可以把这个技术想像为拖拽橡皮圈从  $D(1)$  到  $D(N)$ , 它们的边界由  $D(t)$  和  $W(t)$  确定。总传输数据曲线的斜率为数据传输率。自然地, 我们可以尽量减小斜率(即峰值传输速率), 无论传输数据的速率什么时候发生改变, 在传输规划中它应尽早采取这种措施。

图 16-18 中对这一现象进行了说明。当预取缓冲期处于状态  $a$  时, 服务器开始传输数据。为了避免缓冲器饥饿, 传输率要足够高, 使得在  $c$  处有足够的缓冲。然而, 在该速率下, 缓冲器会在  $b$  处发生溢出, 这样就必须在  $b$  和  $c$  间的某处降低传输率。

在这个范围内, 最近的一个点(使得传输率可变性最小)为点  $c$ 。传输率在  $c$  点减小到一个比较小的值, 并保持这个值一直到点  $d$ , 在  $d$  处缓冲器为空。在此之后, 传输率应继续降低(低于平均码率), 以避免缓冲器溢出, 直到  $e$  点。达到  $e$  点后, 传输率开始上升。

考虑在任意时间段  $[p, q]$ , 令  $B(t)$  表示在时间  $t$  缓冲器中的数据总量。在不引起缓冲区溢出的前提下, 最大常数数据速率可以用下面的  $R_{max}$  表示:

$$R_{max} = \min_{p+1 \leq t \leq q} \frac{W(t) - (D(p) + B(p))}{t - p} \quad (16.13)$$

在同一时间段内不引起缓冲区饥饿的最小传输速率  $R_{min}$  为:

$$R_{min} = \max_{p+1 \leq t \leq q} \frac{D(t) - (D(p) + B(p))}{t - p} \quad (16.14)$$

显然, 传输过程需要满足  $R_{max} \geq R_{min}$ , 否则在  $[p, q]$  时间段内就无法实现以常数传输率传输。该算法可以用于构造一个最佳传输计划, 该计划从  $[p, q=p+1]$  开始, 每次增加  $q$  的过程中重新计算  $R_{max}$  和  $R_{min}$  的值。如果  $R_{max}$  变大, 则在  $[p, q_{max}]$  间建立一个速率为  $R_{max}$  的播放段, 其中  $q_{max}$  为缓

缓冲区最近充满的时刻（在 $[p, q]$ 间达到 $R_{max}$ 的最近的点）。

同样，如果 $R_{min}$ 变小，那么就在 $[p, q_{min}]$ 间建立一个速率为 $R_{min}$ 的播放段， $q_{min}$ 为缓冲区最近一次为空的时刻。

传输率计划充分考虑了网络允许的最大抖动。假使在接收数据的时候没有延迟，那么在 $t$ 时刻， $A(t)$ 字节的数据被接收，它必须不超过 $W(t)$ 。现在假设网络处于最不理想的状态，有 $\delta$ 秒的最大延迟。视频解码会延迟 $\delta$ 秒，则预处理缓冲器中的内容便不能被释放。这样 $D(t)$ 曲线需要被修正为 $D(t-\delta)$ 曲线。图 16-18 解释了上面的观点。这种方法能在给定最大延迟抖动的情况下提供避免缓冲区饥饿以及溢出的方案。

## 16.6 进一步探索

读者可以参考 Steinmetz、Nahrstedt[27]、Wu 和 Irwin[28]、Jeffay 和 Zhang[29]来了解多媒体网络通信方面的更多内容。Wang 等[30]提供了关于视频处理和通信的详细讨论。Orzessek 和 Sommer[15]中有关于 ATM、MPEG-2 及在宽带网络中集成数字视频等方面的内容。

475

在网站上的本章的 Further Exploration 中列出了一些关于多媒体网络通信的优秀资源，诸如：

- ITU-T 建议
- MBone 站点
- RTP、RTSP 和 SIP 的网页
- ATM 的介绍和白皮书
- DVB 的介绍和白皮书

此外，还包括 IETF 中的 RFC 列表：

- 评价可靠多播传输协议的标准
- 多媒体数据的实时传输协议（RTP、RTSP、RSVP）
- VoIP（SIP、SDP 和 SAP）的协议
- Diffserv 与 MPLS

## 16.7 练习

1. 讨论至少两种在基于为任何多媒体数据包都指定 QoS 类别的包交换网络中启用 QoS 路由的方法（它可以应用于任何存储-转发网络）。
2. 给出几个 16.1.3 节中没有提到过的用于特定多媒体应用的优先级传输方法。
3. RTP 在什么时候使用？RTSP 在什么时候使用？将这两个协议合并有没有好处？
4. 重新考虑说明 RSVP 的图 16-4。在 16-4d 中，接收者 R3 发送一个 RESV 的 RSVP 消息给 S1。假设图中给出了整个网络的状态，那么这个路径是为使未来网络的吞吐量最大而保留的吗？如果不是，什么是优化路径？在不修改 RSVP 协议的条件下，给出一个网络节点发现并选择这样一条路径的策略。
5. 浏览网页来了解当前因特网电话的技术进展。
6. 对于交错广播，如果所有 $K$ （ $K \geq 1$ ）个逻辑通道得到的带宽都一样，证明接入时间与 $K$ 无关。
7. 在图 16-17 中指出可用视频传输策略的特点。什么是优化传输计划？
8. 考虑提前传输使媒体流畅的技术，根据算法，如何决定在哪一点来改变数据的传输率？什么是传输率？
9. 重新考虑提前传输使媒体流畅的技术，可以不使用每个视频帧而只考虑压缩视频段中变化很大的帧。你将如何修正这个算法（或视频信息）来支持上述的改进？

476

10. 当服务器与特定客户端之间建立单独的通信通道时, 进行单播传输。
  - (a) 假设视频是 VBR 编码, 同时允许客户端发送反馈, 举出两个用于单播视频传输的方法。
  - (b) 哪个方法更好, 为什么?
11. 多播传输是服务器将单独的媒体流传送给所有的监听多播路由器, 路由器也用这种方法进行传输, 直到有客户端收到媒体流为止。
  - (a) 对于 VBR 视频流, 列举两个用于多播视频传输的方法。
  - (b) 哪个方法更好, 为什么?

提示: 尽管客户端可能为反馈保留了相应的通道, 但如果所有的客户端都像服务器发送反馈, 可能导致网络的拥塞。

## 16.8 参考文献

- [1] H. Eriksson, "MBONE: the Multicast Backbone," *Communications of the ACM*, 37(8): 54–60, 1994.
- [2] M.R. Macedonia and D.P. Brutzman, "MBone Provides Audio and Video across the Internet," *IEEE Computer*, 27(4): 30–36, 1994.
- [3] V. Kumar, *MBone: Interactive Multimedia on the Internet*, Indianapolis: New Riders, 1996.
- [4] K.C. Almeroth, "The Evolution of Multicast: from the MBone to Interdomain Multicast to Internet2 Deployment," *IEEE Network*, 14: 10–20, January/February 2000.
- [5] S. Paul, et al., "Reliable Multicast Transport Protocol (RMTP)," *IEEE Journal on Selected Areas in Communications*, 15(3): 407–421, 1997.
- [6] B. Whetten and G. Taskale, "An Overview of Reliable Multicast Transport Protocol II," *IEEE Network*, 14: 37–47, January/February 2000.
- [7] C. Liu, "Multimedia over IP: RSVP, RTP, RTCP, RTSP," In *Handbook of Emerging Communications Technologies: The Next Decade*, ed. R. Osso, Boca Raton, CRC Press, 2000, 29–46.
- [8] L. Zhang, et al., "RSVP: a New Resource ReSerVation Protocol," *IEEE Network*, 7(5): 8–19, 1993.
- [9] M. Krunz, "Bandwidth Allocation Strategies for Transporting Variable-Bit-Rate Video Traffic," *IEEE Communications Magazine*, 35(1): 40–46, 1999.
- [10] H. Schulzrinne and J. Rosenberg, "The IETF Internet Telephony Architecture and Protocols," *IEEE Network*, 13: 18–23, May/June 1999.
- [11] *Packet-based Multimedia Communications Systems*. ITU-T Recommendation H.323, November 2000 (earlier version September 1999).
- [12] J. Toga and J. Ott, "ITU-T Standardization Activities for Interactive Multimedia Communications on Packet-Based Networks: H.323 and Related Recommendations," *Computer Networks*, 31(3): 205–223, 1999.
- [13] A.S. Tanenbaum, *Computer Networks*, 4th ed., Upper Saddle River, NJ: Prentice Hall PTR, 2003.
- [14] W. Stallings, *Data & Computer Communications*, 6th ed., Upper Saddle River, NJ: Prentice Hall, 2000.
- [15] M. Orzessek and P. Sommer, *ATM & MPEG-2*, Upper Saddle River, NJ: Prentice Hall PTR, 1998.
- [16] U. Varshney, "Multicasting: Issues and Network Support," In *Multimedia Communications: Directions & Innovations*, ed. J.D. Gibson, San Diego: Academic Press, 2001, 297–310.

- [17] G. Armitage, "IP Multicast over ATM Networks," *IEEE Journal on Selected Areas in Communications*, 15(3): 445–457, 1997.
- [18] S. Viswanathan and T. Imielinski, "Pyramid Broadcasting for Video on Demand Service," In *IEEE Conf. on Multimedia Computing and Networking*, 1995, 66–77.
- [19] K.A. Hua and S. Sheu, "Skyscraper Broadcasting: a New Broadcasting Scheme for Metropolitan Video-On-Demand Systems," In *Proceedings of the ACM SIGCOMM*, 1997, 89–100.
- [20] A. Hu, "Video-on-Demand Broadcasting Protocols: A Comprehensive Study," *Proceeding of IEEE IFCOM '01*, 2001, 508–517.
- [21] L. Juhn and L. Tseng, "Harmonic Broadcasting for Video-on-Demand Service," *IEEE Transactions on Broadcasting*, 43(3): 268–271, 1997.
- [22] J.F. Paris, S.W. Carter, and D.D.E. Long, "A Hybrid Broadcasting Protocol for Video on Demand," in *Proceeding of the 1999 Multimedia Computing and Networking Conference MMCN '99*, 1999, 317–326.
- [23] J.F. Paris, "A Simple Low-Bandwidth Broadcasting Protocol for Video-on-Demand," *Proceeding of the 7th International Conference on Computer Communications and Networks (IC3N '98)*, 1999, 690–697.
- [24] D. Saporilla, K. Ross, and M. Reisslein, "Periodic Broadcasting with VBR-Encoded Video," in *Proceeding of IEEE Infocom '99*, 1999, 464–471.
- [25] D. Eager, M. Vernon, and J. Zahorjan, "Minimizing Bandwidth Requirements for On-Demand Data Delivery," *IEEE Transactions on Knowledge and Data Engineering*, 13(5): 742–757, 2001.
- [26] J.D. Salehi, Z.L. Zhang, J.F. Kurose, and D. Towsley, "Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements through Optimal Smoothing," *ACM SIGMETRICS*, 24(1): 222–231, 1996.
- [27] R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications & Applications*, Upper Saddle River, NJ: Prentice Hall PTR, 1995.
- [28] C.H. Wu and J.D. Irwin, *Emerging Multimedia Computer Communication Technologies*. Upper Saddle River, NJ: Prentice Hall PTR, 1998.
- [29] K. Jeffay and H. Zhang, *Readings in Multimedia Computing and Networking*, San Francisco, CA: Morgan Kaufmann, 2002.
- [30] Y. Wang, J. Ostermann, and Y.Q. Zhang, *Video Processing and Communications*, Upper Saddle River, NJ: Prentice Hall, 2002.

## 第 17 章 无线网络

### 17.1 简介

快速发展的计算机和通信技术使普适计算成为了可能。从早期的无线电话到 20 世纪 90 年代出现的手机,直至当今的个人数字助理(Personal Digital Assistant, PDA)、Pocket PC(口袋 PC)和视频电话等,无线通信都是它们的核心技术。无线通信技术启用了个人通信服务(Personal Communication Service, PCS)、个人通信网络(Personal Communications Network, PCN)和个人数字蜂窝(Personal Digital Cellular, PDC)。

从地理上看,无线网络通常被划分成一个个的蜂窝(cell)。蜂窝中的每个移动电话通过接入点(access point)与他人联系,接入点起到了网关的作用。接入点之间通常采用有线、无线或者卫星相连,从而形成核心网络。当一个移动电话用户走出初始接入点的覆盖区域后,需要进行信号的移交(handoff)来维持原来的通信。

在 1985 年,902~928MHz、2.400~2.4835GHz 和 5.725~5.850GHz 三个频段分别被 FCC 指定用于工业、科学和医疗的应用,因此被称为 ISM 频段。

从传统意义上讲,蜂窝的大小通常是以公里为单位衡量的。然而,PCS 的引入带来了分层蜂窝网络的需求,多种层次的蜂窝定义如下:

- **微微蜂窝(picocell)** 每个微微蜂窝最多覆盖 100 米,适用于家庭和办公室中的无线应用程序和设备(如 PDA 等)。
- **微蜂窝(microcell)** 每个微蜂窝最多覆盖 1000 米的范围,在城市和局部区域内可以得到应用,如街上通过无线接入的公用电话。
- **蜂窝(cell)** 每个蜂窝的覆盖范围约为 10000 米,适用于国家内部的网络应用。
- **宏蜂窝(macrocell)** 范围覆盖全球,如卫星电话。

信号的衰减是无线通信(尤其是移动通信)中存在的普遍现象,接收到的信号强度有时会(瞬间)变小。当一个信号通过不同的路径(比如通过楼房、高山或者其他物体的反射)到达接收器时,就会发生多路衰减现象。因为它们在不同的时间和相位到达,信号的多态会互相抵消,这样会导致信号丢失甚至连接中断。尤其在使用更高的数据速率进行通信时,这种问题变得更加严重。

479

#### 17.1.1 模拟无线网络

早期的无线通信网络大部分是用于语音通信的,如电话和语音邮件。第一代(1G)蜂窝电话使用模拟技术和频分多址访问(Frequency Division Multiple Access, FDMA)技术。在这种技术中,通信的每个用户使用一个独立的频率通道。它的标准是北美的高级移动电话系统(Advanced Mobile Phone System, AMPS),欧洲和亚洲的全访问通信系统(Total Access Communication System, TACS)以及北欧移动电话系统(Nordic Mobile Telephony, NMT)。进行数字数据传输的用户需要调制解调器来接入无线网络,常用的数据速率是 9600bps。

以 AMPS 为例,它在 800~900MHz 频段内工作,为双路通信的每一个方向都分配 25MHz,移动站传输(MS 传输)使用 824~849MHz 频段,而基站传输(BS 传输)使用 869~894MHz 频段。每个 25MHz 的频段都分成两个 12.5MHz 的操作频段——A 和 B,每个频段都是 12.5MHz。FDMA



则继续将每个 12.5MHz 操作频段分成 416 个小通道, 每个通道占用 30kHz 带宽。每个 MS 传输通道的频率低于相应的 BS 传输通道频率 45MHz。

类似地, TACS 使用 900MHz 频段, 它最多支持 1320 个全双工通道, 每个通道的频率宽度为 25kHz。

图 17-1 给出了一个可能的用于 FDMA 蜂窝系统的几何分布 (为清楚起见, 第一个蜂窝集群的蜂窝用粗线标出)。一个含有 7 个六边形蜂窝的蜂窝集群是一个覆盖蜂窝区域。只要给这个集群中的每个蜂窝都分配一组唯一的频率通道, 它们之间就不会产生冲突。

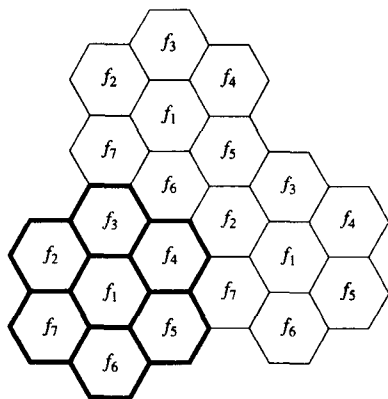


图 17-1 一个可能的 FDMA 蜂窝系统的几何分布, 其集群的大小是 7 个六角形的蜂窝

480

同样的频率通道集合 (如图 17-1 中的  $f_1$  至  $f_7$ ) 会在每个群集中按照均衡模式被重新使用一次, 重复使用系数  $K=7$ 。例如在 AMPS 系统中, 每个蜂窝可以使用的通道数量 (包括控制通道) 为  $416/K=416/7 \approx 59$ 。

在这种配置中, 可以保证两个不同集群中使用相同频率  $f_n$  的用户在地理位置上至少相隔  $D$ , 其中  $D$  是六边形蜂窝的直径。在真空中, 电磁辐射在距离  $D$  之间以  $D^{-2}$  的速度衰减。然而, 在真实的物理空间中, 这种衰减速度可以达到  $D^{-3.5} \sim D^{-5}$ 。这样可以使得 FDMA 方案适用于模拟无线通信, 因为使用相同频率通道的用户之间的冲突可以忽略。

### 17.1.2 数字无线网络

第二代 (2G) 无线网络使用数字技术。除了语音外, 数字数据在文本消息、音频流、电子出版物等应用中的传输量也不断增加。北美的数字蜂窝网络在 1993 年采用两种相互竞争的技术: 时分多址访问 (Time Division Multiple Access, TDMA) 和码分多址访问 (Code Division Multiple Access, CDMA)。在亚洲和欧洲, 使用 TDMA 的全球移动通信系统 (GSM) 在 1992 年被引入。

下面, 我们将首先介绍 TDMA 和 GSM, 然后介绍扩频并分析一下 CDMA。

### 17.1.3 TDMA 和 GSM

顾名思义, TDMA 在多个时间段内创建多个通道, 同时让这些通道共享相同的载波频率。在实际应用中, TDMA 通常与 FDMA 结合使用, 也就是 FDMA 把可分配的全部频谱先分成多个载波频率通道, 再由在时间的维度上继续划分每个频率通道。

GSM 是由欧洲邮电会议 (European Conference of Postal and Telecommunications Administrations, CEPT) 建立的, 它的目标是为创建一个能够处理数百万计的移动用户, 同时提供漫游整个欧洲的服务的移动通信网络建立一个标准。它在 900MHz 频率范围工作, 因此也叫做 GSM 900。在欧洲

也支持 GSM 1800, 该标准是在最初的 GSM 标准上加以修改以使它工作在 1800MHz 频率范围内而得到的。

在北美, GSM 网络使用 1.9GHz 的频率 (GSM 1900)。然而, TDMA 技术的优势是 TIA/EIA IS-54B 和 IS-136 标准的使用。这些标准有时也称做 Digital-AMPS 或者 D-AMPS。IS-54B 在 1996 年被新的 IS-136 标准所取代, IS-136 标准使用数码控制通道 (DCCH) 以及其他增强的用户服务。IS-136 在 800MHz 和 1.9GHz (PCS 频率范围) 两个频率下工作, 它在两个频率下提供相同的数据服务。GSM 与 IS-136 将 TDMA 与 FDMA 结合来使用被分配的频段, 同时使用纯 FDMA 模式 (模拟) 的移动基站来提供向前兼容性。

481

如图 17-2 所示, GSM 900 的上行通道 (移动站发向基站) 使用 890~915MHz 频段, 而下行通道 (基站发向移动站) 使用 935~960MHz 频段。换句话说, 给每个通道分配 25MHz。GSM 的频分继而将每个 25MHz 划分成 124 个载波频段, 每个频段为 200kHz。GSM 的时分将每个载波频段划分为 TDMA 帧; 26 个 TDMA 帧组成一个 120 毫秒的业务信道 (Traffic Channel, TCH), 该通道用来进行语音和数据的载波传播。

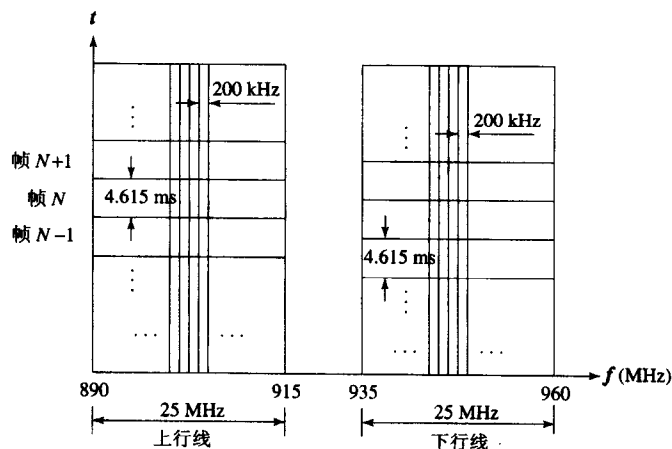


图 17-2 GSM 的频分和时分

每个 TDMA 帧大概是 4.615 毫秒 (即 120/26 毫秒), 它由 8 个时间段组成, 每个时间段长度为  $4.615/8 \approx 0.577$  毫秒。给每个移动站分配一个专用的时间段, 在该时间段中, 移动站可以接收和发送数据。数据的发送和接收并不会同时在一个时间段中发生, 它们被 3 个时间段分隔。

GSM 提供多种数据服务。GSM 用户可以与 POTS、ISDN、包交换或者电路交换公共数据网络中的用户进行数据交换。GSM 同时支持短信息服务 (SMS)。在 SMS 中, 每个文本信息最多有 160 个字符, 可以通过移动电话进行传播。GSM 的一个特点是具有用户识别模块 (SIM), SIM 是一个载有用户电话号码以及访问各种 GSM 服务的智能卡。

通常情况下, GSM 网络是电路交换的, 它的极限数据速率是 9.6kbps。1999 年发展起来的通用分组无线业务 (General Packet Radio Service, GPRS) 支持数据在无线连接上进行包交换, 所以用户是处于 “一直连接” 状态的。它被称为一种 2.5G (介于第二代和第三代之间) 服务。GPRS 理论上的最大速度是 171.2kbps, 这是在 8 个 TDMA 时间段都被一个用户使用的情况下实现的。

在实际的实现中, 单用户的数据流量在 2001 年达到 56kbps。显然, 当网络被多个用户共享的时候, 每个 GPRS 用户的最大的数据速率便会下降。

482

### 17.1.4 扩频和 CDMA

扩频是一种使信号的带宽在传输之前加以扩展的技术。从表面上看, 扩展的信号可能会与背景噪声融合到一起而不易被分辨, 所以它在安全性和健壮性方面具有明显的优势, 可以避免故意的信号干扰 (人为干扰)。

扩频同时既用于模拟信号也适用于数字信号, 因为这两种信号都能被调制并且“扩展”。比如, 早期的无绳电话和蜂窝电话使用的就是模拟信号。然而, 使这项技术在多种无线网络中更受欢迎的应用是数字信号应用, 尤其是 CDMA。

接下来我们将简要介绍两种实现扩频的技术: 跳频和直频。

#### 1. 跳频

跳频 (Frequency Hopping, FH) 是扩频的早期方法。模拟信号跳频的方法是由女演员 Hedy Lamarr[3]在 1940 年二战中发明的。图 17-3 显示了跳频的发送器和接收器的主要组件。

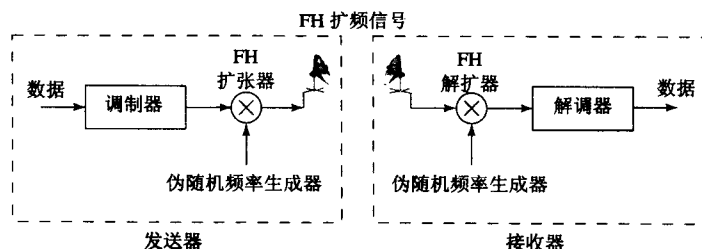


图 17-3 跳频扩频的发送器和接收器

开始, 调制数据 (模拟或者数字) 以产生一些以基频率  $f_b$  为中心的基带信号。因为当前无线应用的数据速率相对较低, 其基带  $B_b$  的带宽也比较小。例如, 如果数据速率是 9.6 kbps, 那么 (根据调制方案) 带宽  $B_b$  不应该超过  $2 \times 9.6 = 19.2 \text{ kHz}$ 。伪随机频率生成器从以兆赫兹 (MHz) 为单位的宽带<sup>①</sup> 中生成一个随机的频率  $f_r$ 。在跳频 (FH) 的扩张器上,  $f_r$  被基带信号调制来产生扩频信号, 它与基带信号的形状相同, 但是具有新的中心频率:

$$f_c = f_r + f_b \quad (17.1) \quad 483$$

因为  $f_r$  在宽带中随机变化, 所以最终信号的  $f_c$  会在宽带中产生相应“跳变”。

在接收端, 处理的过程刚好相反。只要接收端使用相同的伪随机频率发生器产生相同的频率, 就能保证信号被准确地接收和解调。

值得注意的是, 尽管 FH 方法使用宽带扩频方法, 在传输过程中的任意一个给定的时刻, FH 信号只占有频段  $B_b$  的一小部分。

FH 扩频信号的传输在抵御窄带人为干扰方面具有更好的安全性和健壮性, 因为在任何一个窄的频段内, 只有 FH 信号的一个极其微小的部分可能被接收或者受到干扰。

如果跳变的速率低于数据的速率, 则称为慢跳变, 它比较好实现。慢跳变曾在 GSM 中使用, 因为每个具有频率跳变的 TDMA 帧可能在不同的载波频率下传输, 所以慢跳变有助于减少多路径的信号衰减。在快跳变中, 跳变的速率要比数据速率快得多, 所以它在抵御窄带干扰方面要更安全、更有效。

#### 2. 直频

在多接入环境中使用 FH 扩频方案时, 有时会出现多个信号跳变到相同的频率的情况, 这时就

① 频率  $f_r$  的选择是由一个随机数发生器控制的。因为算法是确定的, 所以它并不是真正随机的, 而是伪随机的。

会出现干扰。尽管某些形式的 TDMA 可以减轻这个问题,但它同时对用户的最大数量进行了限制。

无线技术的一个重大突破是码分多路访问 (CDMA) 技术的开发和使用。CDMA 的基础是直频 (Direct Sequence, DS) 扩频。与 FDMA 或者跳频中每个用户在任意时刻占用专门频率不同,多 CDMA 用户在数据的传输过程中可以使用共享宽带通道中相同的 (甚至全部的) 带宽! 一个常用的频段也可以分配给全部蜂窝中的多个用户,换句话说,其重用系数  $K=1$ 。只要能做好用户间的冲突管理,就可以大大提高用户的最大数量。

如图 17-4 所示,对于每个 CDMA 发送器,直频 (DS) 扩展器会接收一个唯一的伪噪声序列。伪噪声序列 (也称为基片码或扩展码) 由一个被称为“基片”的窄脉冲流组成,基片的位宽度为  $T_r$ 。它的带宽  $B_r$  约为  $1/T_r$ 。因为  $T_r$  比较小,所以  $B_r$  比窄带信号的带宽  $B_b$  大很多。

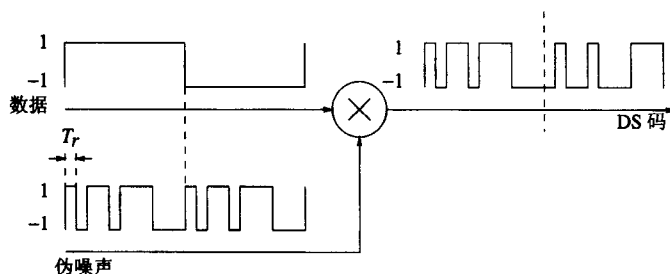


图 17-4 直频扩频

扩展码与输入的数据相乘。当数据位为 1 时,输出的 DS 码与扩展码一致;当数据位为 -1 时,输出的 DS 码与扩展码相反。结果,原始窄带数据的频谱便被扩展,DS 信号的带宽为:

$$B_{DS} = B_r \quad (17.2)$$

解扩展的过程也涉及将 DS 码与扩展序列相乘。只要使用的序列与扩展器中的序列相同,结果信号就会与原始数据相同。图 17-5 显示了 DS 扩展频谱的发送器和接收器的实现。一个与图 17-4 中有微小差别的实现细节是 DS 的扩展器和解扩展器都是模拟设备。在传送到 DS 扩展器之前,数据与扩展序列首先要被调制成模拟信号。

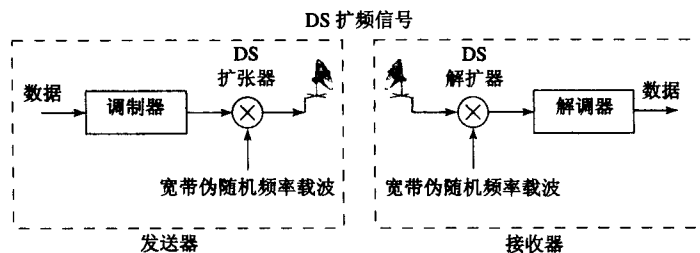


图 17-5 直频扩频中的发送器和接收器

实现 CDMA 多接入的两种方式是:正交码和非正交码。在蜂窝中,动态地给每个移动站分配一个唯一的扩展码,基站使用同样的扩展码来区分和解扩展每个信号。

对于正交的 CDMA,蜂窝中的扩展码是两两正交的[4]。通常情况下使用 Walsh-Hadamard 编码,因为它们具有一种称为正交可变扩展因子 (Orthogonal Variable Spreading Factor, OVSF) 的重要特性。它规定不同长度 (即不同扩展系数) 的 OVSF 码之间仍然是正交的。正交性是我们需要的,因为只要数据被正交码所扩展,它便能在接收端被完好地分离出来。

然而,这种特性也带来了一些问题:当序列不同步时, Walsh-Hadamard 编码可能会出现多个

自相关的尖峰, 所以为了让接收端知道 DS 信号的起始位置, 还需要作一些额外的同步。通常情况下, 在基站中使用全球定位系统 (Global Positioning System, GPS) 可以达到同步的目的。这种特性的另一个缺点是, 正交码都集中在一个很小数量的载波频率中, 这样会导致频谱的利用率低。

非正交码是伪随机噪声 (PN) 序列。PN 序列在 0.5 左右的平均位值, 同时还用一个自相关尖峰来表示序列的开始。这样, PN 序列为自同步的, 它不需要额外的同步工作。常用的一个特殊 PN 序列是 Gold 序列。Gold 序列有 3 个交叉相关的尖峰。

DS 扩展频谱使用了宽带中的全部带宽, 因此, 它在抗干扰方面具有更好的安全性和健壮性。然而, 在多接入的情况下, 因为有多路径衰减、外部蜂窝干扰等原因, 信号仍然会彼此干扰。下面, 我们对 DS 扩展频谱——CDMA 的生存能力进行简要分析。

### 17.1.5 CDMA 分析

当 FDMA 或 TDMA 用于多接入系统中时, 带宽或者时间就会按照最坏的情况 (即所有用户同时并一直接入系统) 来进行分割, 而这种情况在语音通信中几乎不可能出现。CDMA 允许同一通道内的用户共享整个通道带宽。对于每个用户来说, “实际上的噪声” 是其他用户的信号的叠加和, 所以 CDMA 是基于所谓的 “平均情况” 或 “平均干扰” 的。在接收端, DS 输入通过与特定用户指派的扩展码进行相关运算来恢复成原始信号。因此, 只要能维持足够的信噪比, CDMA 的接收质量便有了保证, 就能达到全频复用。

我们假设接收器的热噪声为  $N_T$ , 每个用户接收信号的能量为  $P_i$ 。在基站上接收的源信号之间的干扰为:

$$N = N_T + \sum_{i=1}^{M-1} P_i$$

其中  $M$  为蜂窝中最大的用户数量。

如果我们假设热噪声  $N_T$  可以忽略, 同时每个用户接收信号的能量  $P_i$  均相同, 那么有:

$$N = (M-1)P_i \quad (17.3)$$

每一位接收到的信号能量  $E_b$  是  $P_i$  与数据速率  $R$  (bps) 的比值。

$$E_b = P_i / R \quad (17.4)$$

干扰  $N_b$  为:

$$N_b = N / W = (M-1)P_i / W \quad (17.5)$$

其中  $W$  (Hz) 是 CDMA 宽带信号载波的带宽。

于是我们得到信噪比 (SNR) 为:

$$E_b / N_b = \frac{P_i / R}{(M-1)P_i / W} = \frac{W / R}{M-1} \quad (17.6)$$

整理式 (17.6), 我们可以得到:

$$M-1 = \frac{W / R}{E_b / N_b}$$

或近似表示为:

$$M \approx \frac{W / R}{E_b / N_b} \quad (17.7)$$

式(17.7)是一个很重要的结果。它说明 CDMA 系统的最大容量(即每个蜂窝内用户的最大数量)由两个系数决定:  $W/R$  和  $E_b/N_b$ 。

$W/R$  是 CDMA 带宽  $W$  与用户数据速率  $R$  的比值, 它称为带宽扩展系数或者处理增益。它等于扩展序列中的基片数量。它通常在  $10^2 \sim 10^7$  之间。

$E_b/N_b$  是位级的 SNR。根据 QoS (错误速率需求) 和实现方法 (错误校正方案、多路径衰减的抵制等), 一个数字解调器可以在位级 SNR 为 3dB~9dB 的范围内正常工作。

作为一个例子, 我们假设位级信噪比为 6dB ( $10 \log E_b/N_b = 6\text{dB}$ ), 于是有  $E_b/N_b \approx 4$ 。在 IS-95A 标准中,  $W=1.25\text{MHz}$ ,  $R=9.6\text{kbps}$ 。根据式(17.7)可以得到:

$$M \approx \frac{W/R}{E_b/N_b} \approx \frac{1250/9.6}{4} \approx 32$$

CDMA 系统的最大容量为 32, 这与 AMPS 系统中的容量大致相同。当复用系数  $K=7$ , 带宽为 1.25MHz 时, AMPS 允许的最大通道数量为:  $1250/(30 \times 7) \approx 6$ 。然而, 上面的 CDMA 分析假设相邻的蜂窝间没有信号干扰。如果按照这个假设, AMPS 的复用系数  $K=1$ , 最大的通道数量应为  $1250/30 \approx 42$ 。那么当考虑蜂窝之间的干扰时, CDMA 的容量应该怎么变化呢?

研究表明[5], 相邻蜂窝之间所有用户的干扰仅仅为同一蜂窝内用户间干扰的 60%。因此, 式(17.7)可以引入系数 1.6 (即  $100\%+60\%$ ), 从而改写为反映相邻蜂窝间干扰的表达式:

$$M \approx \frac{W/R}{1.6 \cdot E_b/N_b} \quad (17.8)$$

上述系数 1.6 称为有效复用系数, 因为它在 CDMA 中的作用与 FDMA 系统中的复用系数  $K$  作用相同。很明显, 因为 CDMA 占用所有的带宽, 同时复用系数也比 FDMA 小得多 ( $1.6 < 7$ ), 所以 CDMA 的容量要比 FDMA 的容量大。

根据上面的例子, 我们得到:

$$M \approx \frac{1250/9.6}{1.6 \times 4} \approx 22$$

其容量是 AMPS 的容量的 22/6 倍。

在对 CDMA 分析进行总结前, 必须要指出的是, 我们已经做了一些简化工作。其中某些简化可以增强 CDMA 的性能, 而有些简化会减弱 CDMA 的性能。Viterbi[5]提出了一个对 CDMA 进行完全分析的原则。

- 我们假设每个用户接收的能量  $P_r$  是相同的, 否则当“近的用户”的信号比较强而“远的用户”的信号比较弱的时候, 就会出现“远近问题”, 整个系统便会崩溃。解决这个问题需要在传输端引入一个复杂的能量控制系统。现代 CDMA 能量控制系统每秒 1500 次刷新能量级以保证接收的  $P_r$  都能大体相同。
- 出于严格的能量控制需要, CDMA 网络必须要实现软移交。也就是说, 当移动用户穿过蜂窝的边界时, 因为它处于两个以上基站的范围, 它会立即连接两个以上的通道, 每个通道都对应于不同的基站。在保证发出的信号至少能被一个基站正确接收的前提下, 移动用户选择尽可能减少信号的总能量。这种办法可以使外蜂窝的移动干扰降低到最小。
- 为了减少互斥干扰, 并没有采用全方向的天线, 而是采用具有方向的天线, 把全方向分成若干个扇区。在一个三通扇区蜂窝中, AMPS 的容量减为  $1250/(30 \times 7 \times 3) \approx 2$ 。值得注意的是, CDMA 的容量并不受上述扇区划分的影响。因此, 在划分为扇区的蜂窝中, CDMA

的容量就会更加超过 AMPS 的容量。在上个例子中, 超过的比率为  $22/2 \approx 11$ 。

- 我们已假设每个用户始终需要采用最大的数据速率, 这在实际的应用中是不会出现的。比如, 语音应用只使用系统 35%~40% 的容量。此外, 高效语音编码还可以将网络的容量至少提高 2 倍。

### 17.1.6 3G 数字无线网络

第三代 (3G) 无线服务在多媒体服务方面, 如因特网上的 (低速率) 视频, 起到了重要的作用。3G 的应用包括无线网络浏览、视频邮件、连续媒体点播、移动多媒体、移动电子商务以及远程医疗服务等。与现在的大体上用于室内和专用网络的无线局域网 (WLAN) 不同, 3G 主要用于公共网络。3G 无线网络主要使用宽带 CDMA (Wideband CDMA, WCDMA), 而大部分 2G 无线连接使用 CDMA (如北美使用的 IS-95A) 和 TDMA (其中最常用的是 GSM 和 IS-136)。

488

3G 的标准化过程始于 1998 年, 国际电信联盟在召开的国际移动通信论坛 (IMT-2000) 中提出了无线电传输技术 (Radio Transmission Technology, RTT) 的提案。自那以后, 这个项目便被称为 3G 或者通用移动通信系统 (Universal Mobile Telecommunications System, UMTS)。各国的标准化组织随后采用了该建议, 并加入了它们自己的改进及评价方法提交给 ITU (ITU-R, 国际电信联盟的无线电技术部门)。

值得注意的是, 在相关的规范由其成员 (包含跨国公司) 的各国标准化组织开发的时候, 这些团体仍然倾向于使用一种类似的 WCDMA 技术。为了实现全球标准化及更有效地讨论相关的问题, 第三代伙伴项目 (Third Generation Partnership Project, 3GPP) 于 1998 年年末成立。成立 3GPP 是为了指定 WCDMA 技术的全球标准, 该标准称为通用地面无线电接入 (Universal Terrestrial Radio Access, UTRA) 标准。参与创建 3GPP 论坛的标准化团体有日本的 ARIB、欧洲的 ETSI、韩国的 TTA、日本的 TTC 及北美的 T1, 中国的 CWTS 在 1999 年也加入了该组织。

3GPP 论坛的研究中心关注 WCDMA 的空中接口, 其目标是推进发展 GSM 技术, 同时用来与 GSM MAP 的核心网络进行交互。同时, 得到工业界支持的通信工业协会 (TIA) 也在为 ITU 开发 cdma2000 空中接口, cdma2000 是 IS-95 标准的发展, 它主要用于 ANSI-41 (或 IS-41) 的核心网络。

类似的工作也在亚洲进行, 模仿 3GPP 的例子, 标准化组织决定成立第二个论坛——第三代伙伴项目 2 (3GPP2)。加入该论坛的标准化团体有日本的 ARIB、中国的 CWTS、北美的 TIA、韩国的 TTA 和日本的 TTC。

尽管 3GPP 和 3GPP2 论坛在 WCDMA 空中接口的提案方面具有相似性, 它们仍然在标准上存在分歧。然而, 它们都对创建全球标准感兴趣, 它们都监控彼此的进程同时支持运营商协调组织的建议。这两个论坛达成了称为全球第三代移动通信系统 (G3G) 的一致标准, 该标准有三种模式: 直接扩展 (Direct Spread, DS)、多载波 (Multi-Carrier, MC) 和时分双工 (Time Division Duplex, TDD), 其中 DS 和 TDD 模式被 3GPP 指定用于 WCDMA 中, 而 MC 模式被 3GPP 指定用于 cdma2000 中。所有的空中接口 (所有模式) 可以在两种核心网络中使用。在 1999 年年末, ITU-R 发布了 IMT-2000 规范, 其大部分内容遵循协调后的 WCDMA 标准。

3G 无线多媒体服务要依靠手机技术的快速发展。新一代手机应该支持视频, 具有更好的软件 and 用户接口, 同时电池寿命应该更长。

从支持电路交换通道上数字通信的 2G 无线网络到同时支持电路交换和包交换通道上高速率数字通信的 3G 网络的移植 (或进化) 的路径已经被指定。进化的路径中有一个简单且实现成本比较低 (网络内部结构的改变较小) 的中介步骤, 该步骤称为 2.5G (2.5 代), 它具有增强数据速

489

率和数据包服务（也就是说，在 2G 的网络中增加了包交换）。表 17-1 概括了使用 IS-41 核心网络（在北美）和 GSM MAP 核心网络（在欧洲等地区）的 2G、2.5G 和 3G 标准（包括已经开发的和即将开发的）。

表 17-1 2G 到 3G 无线网络的进化

	ANSI-41 核心网络	峰值数据率 <i>R</i>	载波频段 <i>W</i>
2G	cdmaOne (IS-95A)	14.4 kbps	1.25 MHz
2.5G	cdmaOne (IS-95B)	115 kbps	1.25 MHz
3G	cdma2000 1X	307 kbps	1.25 MHz
3G	cdma2000 1xEV-DO	2.4 Mbps	1.25 MHz
3G	cdma2000 1xEV-DV	4.8 Mbps	1.25 MHz
3G	cdma2000 3X	>2 Mbps	5 MHz
	GSM MAP 核心网络	峰值数据率 <i>R</i>	载波频段 <i>W</i>
2G	GSM (TDMA)	14.4 kbps	1.25 MHz
2.5G	GPRS (TDMA)	170 kbps	1.25 MHz
3G	EDGE (TDMA)	384 kbps	1.25 MHz
3G	WCDMA	2 Mbps	5 MHz

### 1. IS-95 的演化

IS-95A 和 IS95-B 被认为是 cdmaOne，它们基于 IS-41 核心网络，并使用窄带 CDMA 空中接口。同样的，所有的开发都遵循着把现有的 CDMA 框架扩展到 3G（宽带 CDMA）使其具有向前的兼容性的原则。这里涉及性价比的问题，所以它主要得到了工业界的支持并被其迅速的采用。IS-95A 是一种 2G 技术，它只支持电路交换，数据传输的最大速率为 14.4kbps；它的一个扩展是 IS-95B（2.5G），它支持包交换，同时能达到最大 115kbps 的传输速度。

IMT-2000 MC 模式最初称为 cdma2000，它能在 IMT 频谱的各种频段（450、700、800、900、1700、1800、1900 和 2100MHz）下工作。为了简化 cdma2000 的部署，过渡的框架由 4 个阶段组成，每个阶段兼容前面的阶段，同时与 cdmaOne 兼容。

cdma2000 1X（或 1X RTT）规范也称为高速率包数据空中接口规范，它提供了增强的服务，它的传输尖峰速率可以达到 307kbps，平均速率可以达到 144kbps。这个空气接口提供的容量是 IS-95B 的 2~3 倍。1X 表示它为 cdmaOne 占用 1 倍带宽的通道——每个通道 1.25MHz 载波带宽。和 IS-95 空中接口一样，其基片的速率为 1.2288Mcps（兆基片每秒）。

cdma2000 的下一步部署是 cdma2000 1xEV（EV 代表进化），它可以分成两个阶段。尽管应首先考虑与 ANSI-41 兼容，但它的空中接口试图同时支持 ANSI-41 和 GSM MAP 网络。第一个阶段称为 1xEV-DO（只限数据，Data Only），该阶段支持的最高数据传输速率为 2.4Mbps。语音通信通过单独的通道传输。第二个阶段称为 1xEV-DV（数据和语音，Data and Voice），它使得 1xEV 的接口同时支持语音通信。它还承诺实现更高的数据传输速率（最高达到 4.8 Mbps）。

490

3G 的最后阶段就是在 IMT-2000 中推荐 MC 模式。它被看作是 cdma 2000 3X（或 3X RTT），因为它使用 5MHz（3×1.25MHz 的信道）载波频谱来传送 2~4 Mbps 的峰值速率。芯片速率也是三倍于 3.686 Mcps。

典型的 3G 数据库是 2 Mbps（室内固定应用）和 384 kbps/128 kbps（慢/快速移动用户）。

### 2. GSM 的演化

GSM 无线接入网络（RAN）使用 GSM MAP 核心网络。IMT-2000 的 DS 和 TDD 模式就是基



于为 GSM MAP 网络开发的 WCDMA 技术的。GSM 是基于 TDMA 的技术, 因此它与 WCDMA 的兼容性比 IS-95 差。因此, 3G 的 WCDMA 标准并没有与目前的 GSM 网络兼容。此外, 每次向 3G 的过渡都需要移动站对其他操作模式的支持。

GSM 是只支持电路交换通信的 2G 网络。通用分组无线业务 (GPRS) 是一个 2.5G 的增强技术, 它支持包交换和更高的数据速率。和 CDMA 2000 1X 一样, EDGE (全球进化增强数据率或增强数据 GSM 环境) 支持的最高数据速率是 GSM 和 GPRS 的 3 倍。EDGE 仍然是一个基于 TDMA 的标准, 它主要用于 GSM 向 WCDMA 的进化。然而在 IMT-2000 中, 它被定义为用于单载体模式 (IMT-SC) 的 UWC-136, 这也是一个 3G 的解决方案。使用新的调制和无线电技术, 它能达到最高 384kbps 的数据速度, 从而优化可用频谱的使用。

最终, 3G 技术 (也称为 3GSM) 会根据 IMT-2000 建议的 WCDMA 模式进行修改。WCDMA 具有两种操作模式: 直频 (DS), 也称为频分双工 (FDD) 和时分双工 (TDD)。FDD 模式用于在数据上行通道和下行通道中采用不同的频率时分配的成对的频谱中。对于没有出现频率对的网络而言, 数据在上行和下行通道中进行传播应该采用同一个频率。这是使用时间段来完成的, 上行和下行传输在不同的时间段内进行。在移动端, 它还需要一个比 TDMA 更复杂的时间控制系统。

WCDMA 的空中接口与窄带 CDMA 空中接口的主要区别如下:

- 为了支持高达 2Mbps 的码率, 需要为其分配更宽的通道带宽。WCDMA 的通道带宽为 5MHz, 而 IS-95 和其他早期标准的带宽为 1.25MHz。
- 为了充分利用 5MHz 的带宽, 还需要有更高基片速率的更长的扩展码。WCDMA 指定的基片速率为 3.84Mcps, 而 IS-95 的指定基片速率为 1.2288Mcps。
- WCDMA 支持可变码率, 其范围从 8kbps~2Mbps。这是使用可变长度的扩展码和 10 毫秒的时间帧来达到的。在 10 毫秒的时间帧中, 用户数据速率保持不变, 但是可以从一个数据帧转换到另一帧——这便是按需带宽 (Bandwidth on Demand, BoD)。
- WCDMA 基站使用带有 Gold 编码的异步 CDMA, 这样便避免了 IS-95 的基站中需要使用 GPS 来保证时间同步的问题。WCDMA 的基站可以变得更小, 成本更低, 进而可以被放置在室内。

491

### 17.1.7 无线局域网

由于有多种多样的语音和数据应用, 一直以来, 无线 WAN (广域网) 都很流行。越来越多的膝上型电脑用户对无线 LAN (局域网) 产生了极大的兴趣。不仅如此, 最近出现的普适计算[6]带来了无线 LAN (WLAN) 的新浪潮。

#### 1. IEEE 802.11

IEEE 802.11 是 IEEE 802.11 工作组最早开发 WLAN 标准。它规定了位于半径为几百英尺的局部区域之内的无线连接的介质访问控制 (MAC) 和物理 (PHY) 层。PHY 既支持跳频 (FH) 扩展频谱, 也支持直频 (DS) 扩展频谱。所使用的 ISM 频段为 2.4GHz。此外, 它也能支持 (散射的) 红外线来进行 10~20 米范围内的户内通信。

WLAN 可以作为有线 LAN 的替代物或者一个扩展。和以太网类似, 802.11 的基本访问控制方法是载波侦听多路访问/冲突避免 (CSMA/CA)。802.11 支持的数据传输率为 1Mbps 和 2Mbps。

802.11 标准也解决了下列重要问题:

- 安全 加强了认证和加密, 因为 WLAN 很容易被侵入。
- 能源管理 在没有传输的时候节约能源, 并控制休眠和唤醒。

- 漫游 允许不同接入点对基本消息格式的吸收。

## 2. IEEE 802.11b

IEEE 802.11b 是对 802.11 的一个增强。它依然使用 DS 扩展频谱,并在 2.4GHz 频段上工作。在新的技术,特别是在补码键控 (CCK) 调制技术的帮助下,除了原来的 1Mbps 和 2Mbps,它还能支持 5.5Mbps 和 11Mbps,它的功能已经与以太网不相上下了。

在北美,802.11 和 802.11b 分配的频谱为 2.400 GHz~2.4835GHz。不考虑传输率 (1Mbps、2Mbps、5.5Mbps,或者 11Mbps),DS 扩展频谱通道的带宽为 20MHz。三个非重叠的 DS 通道能够同时使用,允许一个局部区域内最多可以有 3 个接入点。

IEEE 802.11b 已经得到公众的吸收,并出现在所有 WLAN 中,包括大学校园、机场、会议中心等。

## 3. IEEE 802.11a

IEEE 802.11a 在 5GHz 的频段上工作,并支持 6Mbps~54Mbps 的传输率。它使用正交频分复用 (OFDM),而不是 DS 扩展频谱。它允许 12 个非重叠的通道,因此一个局部区域内最多可有 12 个接入点。

因为 802.11a 在更高频段上 (5GHz) 工作,与 802.11 和 802.11b 相比,它面对的射频 (RF) 干扰 (例如来自无绳电话) 较少。由于具有较高的传输率,它可以很好地支持 LAN 环境中各种多媒体应用。

高性能无线电局域网 (HIPERLAN/2) 是 IEEE 802.11a 在欧洲的一种同属。它也工作在 5GHz 的频段上,能够提供最高 54Mbps 的传输率。Wesell[2]中有关于 HIPERLAN 的详细描述。

## 4. IEEE 802.11g 和其他

IEEE 802.11g 是 802.11b 的扩展,是对在 2.4GHz 频段下获得最高 54Mbps 传输率的一种尝试。和 802.11a 一样,这里使用了 OFDM 而不是 DS 扩展频谱。然而,802.11g 仍然会遇到比 802.11a 更高 RF 干扰,就像在 802.11b 中一样,它在局部区域内最多只能有 3 个接入点。

IEEE 802.11g 向下兼容 802.11b,这给 802.11g 网络上所有的 802.11b 和 802.11g 用户都带来了极大的开销。

另外 6 个 802.11 标准正在开发之中,用于处理 WLAN 不同的方面。本章的 17.4 节有这些标准的 WWW URL。特别的,802.11e 处理关于 QoS 的 MAC 增强,特别是语音和视频的优先传输。

## 5. 蓝牙

蓝牙 (以 10 世纪的丹麦国王 Harold Bluetooth 命名) 是一种用于短距离无线通信的新协议。特别的,它能够代替连接移动和固定的计算机和设备的电缆。它使用位于 2.4GHz ISM 频段的 FH 扩展频谱和一个全双工的信号,这个信号在以 1MHz 为间隔的 79 个频率之间跳动,跳动的频率为每秒 1600 跳。蓝牙支持电路交换和包交换。它最多支持 3 个语音通道 (每个通道都是对称的 64kbps) 和多于一个的数据通道 (每个通道都是对称的 400kbps)。

蓝牙联盟的网站 (www.bluetooth.com) 提供了详细的核心规范,包括在蓝牙环境中使用无线应用协议 (WAP) 的描述。例如,在 “briefcase trick” 中,用户的移动电话可以周期性地与他的膝上型电脑通信,这样就可以从手持电话上查看 e-mail 而不需要打开公文包。

Sony 的一些便携式摄像机已经内置了蓝牙接口。这便可以将移动的或者静止的图片发送到 PC 上,或者通过装配有蓝牙的移动电话直接发送到网络上去,在 10 米之内,传输速率可以超过 700kbps。这种便携式摄像机甚至能够用于浏览 WWW 和发送带有 JPEG 或者 MPEG-1 附件的 e-mail。

## 17.2 无线电传播模式

无线电传输通道具有的工程难题比有线通道更多。在这一节里,我们将简要介绍最常用的无线电通道模型来了解造成位/帧错误的原因,并对位错误进行分类,了解错误的数量,以及是否是猝发的情况。

很多效应会造成无线电信号在接收端的质量下降(这里所指的并不只是噪声)。这些效应可以分为短程效应和长程效应。相应地,多径衰减模式适用于小规模衰减通道,而路径损耗模型则适用于长程的大气衰减通道。

493

对于户内通道,无线电信号的强度都比较低,狭小空间内物体也比较多,有的还在运动。因此,多径衰减是信号质量下降的主要因素,由此可以建立衰减模型。在这种环境里,传输的信号被分开通过多个路径到达接收者,每个路径都有自己的衰减程度、相位延迟和时延。

多径衰减模型随机确定接收到的信号的振幅,根据接收者添加的信号是破坏相加还是构造相加,确定的振幅是不同的。信号衰减的原因包括反射、折射、散射和衍射(主要由移动物体引起)。

户外也有折射、衍射和散射效应,多数是由地面和建筑物引起。然而,长程通信中占主导地位的是大气衰减。根据频率的不同,无线电波能够穿透电离层(> 3 GHz)并建立起视距(LOS)通信,而频率稍低的电波被电离层和地面反射,或者沿着电离层传播到接收端。高于 3 GHz 的频率(要保证卫星通信穿透电离层,这是必需的)会经历气体的衰减,氧气和水(水蒸气 and 降雨)是主要影响因素。

### 17.2.1 多径衰减

衰减模型尝试着对接收端添加的信号振幅进行建模。信号的多普勒扩散定义为频谱上信号强度的分布(信号在特定的频率带宽上进行调制)。当信号的多普勒扩散足够小的时候,信号是一致的,也就是说,接收端只有一个可辨识的信号。窄带信号通常是这种情况。然而,如果信号是宽带的,不同频率的信号有不同的衰减路径,在接收端可以观测到多个可不同的分时信号路径。对于窄带信号,最常用的模型是 Rayleigh 衰减(瑞利衰减)和 Rician 衰减(莱斯衰减)。

Rayleigh 模型假定到接收者的无数个无视距(Line-of-Sight, LOS)信号路径,建立起关于接收信号振幅  $r$  的概率密度函数  $P_r$ :

$$P_r(r) = \frac{r}{\sigma^2} \cdot e^{-\frac{r^2}{2\sigma^2}} \quad (17.9)$$

这里  $\sigma$  是概率密度函数的标准差。虽然信号路径的数目通常都不大,但 Rayleigh 模型仍然对路径数大于 5 的情况提供了一个很好的近似。

假定了一个视距的更常用的模型就是 Rician 模型。它定义了一个  $K$  因子作为信号强度和散射强度的比值。也就是说,  $K$  就是 LOS 的信号比其他路径强的倍数。Rician 概率密度函数  $P_c$  为

$$P_c(r) = \frac{r}{\sigma^2} \cdot e^{-\frac{r^2}{2\sigma^2} - K} \cdot I_0\left(\frac{r}{\sigma} \sqrt{2K}\right), \text{ 其中 } K = \frac{s^2}{2\sigma^2} \quad (17.10)$$

像前面一样,  $r$  和  $\sigma$  分别为信号的振幅和标准差,  $s$  为 LOS 信号强度。 $I_0$  是一个修改过的 0 阶的第一类贝塞尔函数。注意,当  $s=0$  ( $K=0$ ) 的时候, LOS 并不存在,因此模型简化为一个 Rayleigh 分布。当  $K=\infty$  时,模型反映了加性高斯白噪声(AWGN)状况。图 17-6 展示了当  $K$  因子分别为 0、1、3、5、10 和 20, 标准差  $\sigma=1.0$  时的 Rician 概率密度函数。

对于宽带信号, 衰减路径更多是通过经验获得。其中一种方法是将振幅建模为所有路径上的总和, 每一个路径都有随机的衰减。对于一个封闭的环境, 路径数目可以为 7 (6 面墙和 LOS), 对于其他环境, 路径数可以取得更大。另一种技术是通过测量通道的冲激响应来为通道衰减建模。

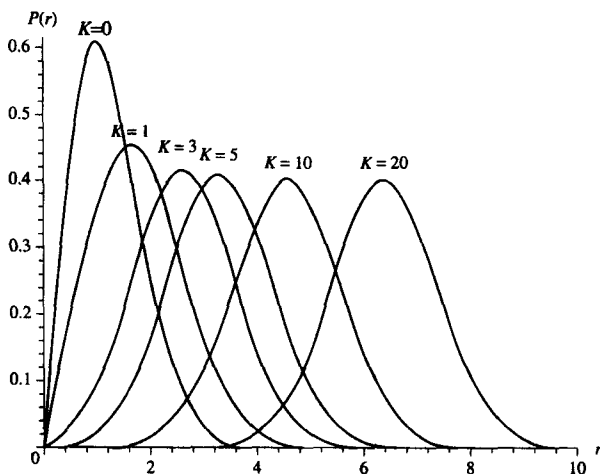


图 17-6  $K$  因子为 0、1、3、5、10 和 20 的 Rician PDF 图

CDMA 系统中使用一种相似的技术, 这种技术也在 cdma2000 中提出, 也被加入到 WCDMA 作为最后一致同意的方案的一部分。CDMA 站 (既指移动站也指基站) 具有耙式接收器, 其实是多个 CDMA 无线电接收器, 这些接收器调谐为具有不同相位和振幅的信号, 把划分为不同可辨识路径的 CDMA 传输重新组合在一起。将每个耙式接收器中的信号累加起来以获得更好的 SNR。为了把耙式接收器调谐到适当的衰减路径, CDMA 系统通过一个特殊的导频通道发送一个导频信号, 耙式接收器做出相应的调整来识别每个衰减路径上的符号。

### 17.2.2 路径损耗

对于长程通信, 信号的损失主要是衰减。LOS 传输的空闲空间衰减模型与距离的平方 ( $d^2$ ) 成反比, 并由 Friis 方程给出

$$S_r = \frac{S_t G_t G_r \lambda^2}{(4\pi^2) d^2 L} \quad (17.11)$$

$S_r$  和  $S_t$  是接收到和传输的信号强度,  $G_r$  和  $G_t$  是天线增益因子,  $\lambda$  是信号的波长, 而  $L$  为接收端的损耗。然而, 从中可以看出, 如果假设有地面反射, 衰减会增加到与  $d^4$  成正比。

另一种常见的介质规模 (城市大小) 模型是 Hata 模型, 它是基于东京的 Okumura 路径损失数据根据经验推导出来的。以 dB 为单位的路径损失方程的基本形式由下式给出:

$$L = A + B \cdot \log_{10}(d) + C \quad (17.12)$$

这里,  $A$  是频率和天线高度的一个函数,  $B$  是一个环境函数,  $C$  是和载波频率有关的一个函数。同样,  $d$  是发送端到接收端的距离。

卫星模型的衰减主要由降雨引起。因此, 气象降雨密度图可以用在这个领域中。衰减可以根据该区域在给定日期的降雨量计算得到。

### 17.3 无线网络上的多媒体

我们已经学习了当前2G网络到未来高容量3G网络的演变,但是3G网络真的是有必要的吗?无线网络上的多媒体显然需要更高的带宽。暗含的多媒体应用包括从网站浏览、流视频、视频会议、协同工作、幻灯片演示到增强的路边信息亭和可供司机下载的GPS地图。

在这一节中,我们主要关注在无线网络上稳定地传输视频,比如为视频会议发送视频。这种应用在3G手持设备上将会大放异彩,因为它是语音通信的一种自然的扩展。

由于无线数据传输可能出现大量的数据丢失和失真,因此对错误具有恢复能力和错误修复就成为我们需要关注的话题了。因此,我们在这一节介绍一些关于同步损失、错误恢复熵编码、错误隐藏和前向纠错(FEC)的概要描述,虽然其中的大部分技术也可以应用到其他类型的网络中。

设计多媒体传输(特别是视频传输)的时候,要考虑无线手持设备的一些特性。首先,手持设备的大小和电池时间限制了设备的处理能力和内存容量。因此,编码和解码的复杂度必须相对较低。当然,较小的设备的一个优点就是可以接受较低分辨率的视频,这有助于减少处理时间。

496

其次,由于内存限制和使用无线设备的原因,以及计费过程,实时通信很有可能是必需的。看到视频之前有太长的延迟是让用户无法接受的。

最后,无线通道比有线通道的干扰更大,根据环境状况有不同的数据丢失现象。无线通道的码率也更加有限,尽管3G的码率更适合于视频。这意味着虽然要应用大量的位保护,但是也必须考虑编码效率。错误恢复编码是很重要的。

3G标准规定,视频应该是标准兼容的。不仅如此,大多数公司会应用标准来开发产品,这是为了保证移动设备和网络的互操作性。适合无线网络使用的视频标准包括MEPG-4和H.263和它的变体,因为它们所需的码率较低。

3GPP2小组为无线视频会议服务定义了如下QoS参数[7]。无线部分规定的QoS参数相对于端到端传输所需的参数更加严格。而3GPP中多媒体传输的QoS需求基本上是一致的[8]。

- **同步** 视频和音频应在20毫秒内保持同步。
- **吞吐量** 最小的视频码率为32kbps,也支持128kbps、384kbps和更高的速率。
- **延迟** 端到端传输的最大延迟为400毫秒。
- **抖动** 最大的延迟抖动(平均的延迟和95%的延迟分布之间的最大差别)为200毫秒。
- **错误率** 对于电路交换传输来说,视频会议系统应该能够容忍 $10^{-2}$ 的帧错误或 $10^{-3}$ 的位错误率。

接下来,我们将要讨论视频序列面对位错误时的脆弱性和提高适应错误能力的方法。

#### 17.3.1 同步损失

视频流既可以在分包后通过包交换通道传输,也可以作为连续的位流在电路交换通道上传输。在任何一种方式下,包丢失或者位错误都会降低视频质量。如果位丢失或者包丢失只是出现在视频空间和时间的一个局部区域,那么这种丢失是可以接受的,因为一帧只会显示很短的一段时间,并且小的错误并不会引起注意。

然而,数字视频编码技术涉及变长编码,帧是按照不同的预测和量化级别进行编码的。遗憾的是,当一个包含变长信息数据的包(比如DCT的系数)损坏,那么这个错误(如果不限的话)将会在整个流上传播。这种现象叫做解码同步损失。即使解码器能够因为无效的编码符号或系数超出范围而检测到错误,它仍然不能找到下一个开始解码的位置[9]。

497

正如我们在第10章中所学习到的,按照标准协议层编码的视频不会出现这种位流完全损失的情况。图像层和块组(GOB)层或者切片头含有同步标记,可以保证解码的重同步。例如,H.263

位流有四个层——图像层、GOB 层、宏块层和块层。图像层从唯一的 22 位图像开始编码 (PSC) 开始。最长的熵编码符号为 13 位, 因此 PSC 可作为一个同步标记。GOB 层用于几个块而不是整个帧后的同步。块组开始编码 (GBSC) 为 17 位长, 也能够作为同步标记<sup>①</sup>。宏块和块层并不包含唯一的开始编码, 因为认为它们会引起高的开销。

H.261 后的 ITU 标准 (例如 H.263、H.263+ 等) 支持片段结构模式而不是 GOB (H.263 Annex K), 其中, 片段组按照块的编码位长而不是块的数目聚集在一起。其目的是为了在排开这些片段头的时候保证它们之间的距离在已知范围之内。这样做的话, 当一个位流错误看上去像一个同步标记, 如果这个标记并不是片段头应该在的位置, 它将被丢弃, 就不会发生错误的重同步。

由于片段需要将一定数目的宏块组合在一起, 而宏块是使用 VLC 进行编码的, 因此要让所有的片段大小一致是不可能的。不过, 这里是存在一个最小距离的, 在这个距离之后, 下一个扫描到的宏块将加入到一个新的片段中。我们知道, 宏块和运动向量中的 DC 系数的编码方式是不一样的。因此, 即使一个宏块被破坏且解码器确认了下一个同步标记的位置, 它也不一定能够对这个流进行解码。

为了减轻这个问题, 片段也重置了空间预测参数, 跨越片段边界的不同编码方式是不允许使用的。ISO MPEG 标准 (和 H.264) 规定片段并不一定具有相近的位长, 这样一来也无法保证不出现错误的标记。

除了同步损失, 我们也应当注意到, 预测参考帧中的错误比不用于预测的帧中错误引起的信号质量损失更多。也就是说, I 帧中的帧错误对视频流质量的损害要大于 P 帧或者 B 帧中的错误。类似的, 如果视频是可扩展的, 那么基础帧中的错误对视频流质量的损害要大于增强帧中的错误。

MPEG-4 定义另外一种错误恢复工具, 它们可用于噪声或者无线通道环境下的编码。它们是对片段编码和可逆变长编码 (RVLC) 的补充[10, 11]。为更加有助于同步, 一种数据划分方案可以将头信息、运动向量和 DCT 系数包括和分离到不同的包中, 并在它们之间加上同步标记。我们随后将看到, 这样的方案对非对称保护的前向纠错 (FEC) 方案也是很有好处的。

另外, 还可以使用一种帧内自适应刷新模式, 在这种模式下, 每个宏块基于其运动独立地编码为帧间或者帧内的块, 从而有助于错误隐藏。运动快的块需要的刷新频率更高。也就是说, 通常会按照帧内模式编码。同步标记辨认起来会更加容易, 特别适合处理能力有限的设备, 比如手机和移动设备。

对于交互式应用, 如果编码器可以使用后向通道, 那么就可以使用一些额外的错误控制技术, 这些技术可归纳为发送端-接收端反馈类别。根据当前可用的带宽, 接收端可以请求发送端降低或者提高视频的码率 (传输速率控制), 这可以处理由于拥塞而引起的包丢失。如果流是可扩展的, 那么它也能够控制增强层。

除此之外, H.263+ 的 Annex N 规定接收端能够注意到参考帧中的错误, 并请求编码器使用其他参考帧来预测——解码器已经正确重建的参考帧。

以上的技术能够在无线实时视频应用 (比如视频会议) 中使用, 因为无线蜂窝通信在必要情况下可以支持一个后向通道。然而, 很明显, 不用后向通道 (它会减少上行链路中的多路访问干扰) 开销会比较小。

### 17.3.2 错误修复熵编码

GOB、片段和同步标记的主要目标都是尽快在出现错误后重新建立起解码器的同步。在

<sup>①</sup> 同步标记一般都比最小需求要大, 以防止位错误将一些位变得和同步标记一样。

H263+的 Annex N 中, 片段的修复效果更好, 因为它们进一步限制了流可以在哪里进行同步。然而, 另外一种称为错误修复熵编码 (EREC) 的算法在每一个宏块后都能够达到同步, 而并没有片段头或者 GOB 头那样的开销。这种算法之所以叫做 EREC, 是因为它使用了熵编码变长宏块, 并且以一种错误修复的方式来重新排列它们。另外, 它可以适度地降低级别。

EREC 使用一些块进行编码后的位流, 并把它们重新排列, 以使得所有块的开头都按照固定距离分开。虽然这些块的大小是任意的, 也可能是我们希望同步的任何媒体, 但接下来要描述的是视频的宏块。算法的流程如图 17-7 所示。

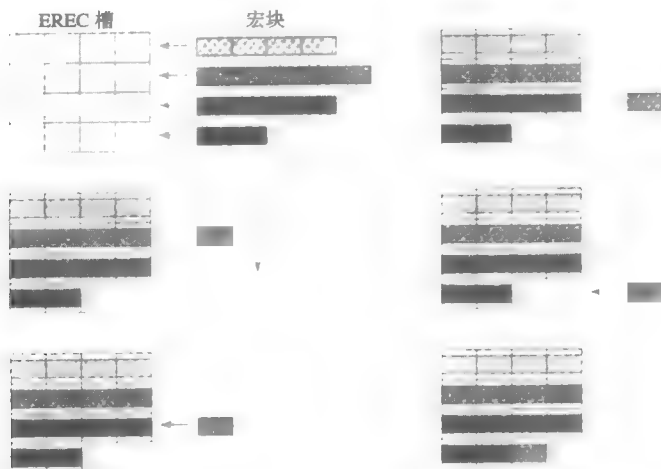


图 17-7 使用 EREC 编码宏块的例子

最初, 固定长度的 EREC 槽 (行) 分配到的总位长等于 (或超过) 所有宏块的总位长。槽的数目与宏块的数目相等, 只是宏块有变化的位长, 而槽的位长是固定的 (大约为所有宏块的平均位长)。如图所示, 当总的位数没有按照槽的数目等分的时候, 最后一个 EREC 槽 (行) 会比较短。

令  $k$  为宏块的数目, 这个数目与槽的数目相等,  $l$  为所有宏块的总位长,  $mbs[]$  为宏块,  $slots[]$  为 EREC 槽, 那么对宏块编码的过程如下所示。

#### 过程 17-1 EREC\_Encode

```

BEGIN
  j = 0;
  Repeat until l = 0
  {
    for i = 0 to k - 1
    {
      m = (i + j) mod k;
      // m is the macroblock number corresponding to slot i;
      Shift as many bits as possible (without overflow) from mbs[i] into slots[m];
      sb = number of bits successfully shifted into slots[m] (without overflow);
      l = l - sb;
    }
    j = j + 1;    // shift the macroblocks downwards
  }
END

```

宏块被移到相应的槽, 直到宏块所有的位都已经被分配或者宏块剩下的位装不进槽中为止。这时, 宏块下移, 过程重复执行。

解码器工作过程则是相反的, 另外它必须检测宏块什么时候已经读满。当所有的 DCT 系数都已经被解码时, 它通过检测宏块的结束 (或者一个块的结束码) 来实现这个功能。图 17-8 展示

了对宏块进行解码的过程的例子，这些宏块是使用图 17-7 所示的 EREC 进行编码的。

槽中数据的传输次序是行为主的，也就是说，槽 0 中的数据最先被发送，接下来是发送槽 1 中的数据，依此类推，从左到右。很容易看出，这个技术对错误具有恢复能力。无论在什么地方出现损坏，即使在宏块的开始，我们仍然知道下一个宏块从何处开始——它与前一个宏块之间的距离是固定的。在这种情况下，并不涉及同步标记的使用，因此 GOB 层或者片段也是不必要的（虽然我们仍然希望能够限制错误在空间上的传播）。

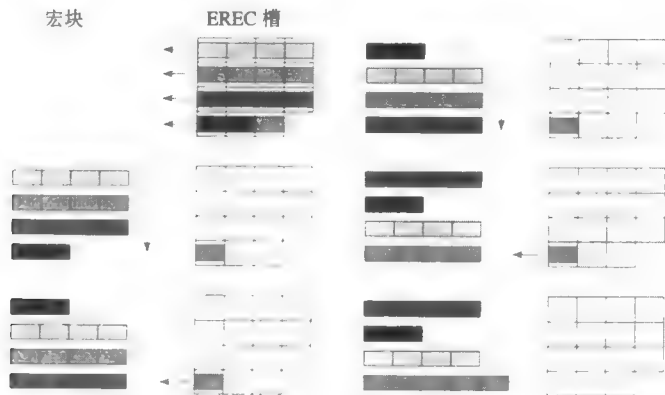


图 17-8 使用 EREC 对宏块进行解码的例子

若宏块使用数据划分技术（比如前面所描述的 MPEG-4 中使用的技术）和位平面划分进行编码，当接收有效数据时，位流中的错误对有效数据产生的破坏相对比较小。显然，槽末端的位发生错误传播的可能性要大于槽起始端的位。一般情况下，这也会减轻非划分编码方式下的视觉退化。在错误严重的情况下，这能够取得比较好的降级效果。

### 17.3.3 错误隐藏

尽管人们很努力地减少错误的发生率并降低错误的影响，但错误仍然让人烦恼。因此，人们引入了错误隐藏技术来估计解码器端损失的数据。

501

许多错误隐藏技术应用在空间、时间、频域上，或者它们的组合。所有的技术都使用时间上邻近的帧或者空间上邻近的宏块。传输流编码器将视频包交织起来，这样万一出现包丢失，错误不会发生在同一个地方，丢失的数据也可以根据邻近数据估计出来。

错误隐藏对于无线视频通信来说是很必要的，因为它的错误率比有线通道的要高，甚至可能比应用合适的位保护来传输的错误率要高。不仅如此，根据不同的移动性或者天气状况，错误率的变化会很频繁。由于数据丢失或者接收错误而引起的解码错误在分辨率有限和屏幕较小的设备上更容易被发现。如果宏块比较大，以便于在低的无线码率下获得好的编码效率，这种情况会特别明显。

接下来是错误隐藏技术的一个总结（参考[12]来获得更详细的信息）。

#### 1. 处理丢失的宏块

当 DCT 块损坏但正确接收到运动向量的时候，有一种简单而常用的隐藏技术可以使用。假定没有预测错误，丢失块的系数可以从参考帧中估计出来。由于运动补偿视频的目的就是将预测错误减到最少，因此这是一个适当的假定。这样，丢失块可以使用参考帧中的块来临时代替。

如果视频是可扩展的，我们甚至能够获得更好的效果。在这种情况下，我们假定基础层已正确接收到，并且它包含运动向量和最重要的基础层系数。这时，对于增强层中丢失的宏块，我们



可以使用基础层中的运动向量来代替增强层中的 DCT 系数,而解码方法不变。由于重要性较低的系数已估计出来(例如高频系数),因此即使是估计值由于预测错误而不够精确,隐藏的效果仍然比非扩展情况要好。

如果运动向量的信息也已经损坏,只有当运动向量通过其他隐藏技术(接下来将要讨论)估计出来时,才能够使用这个技术。运动向量的估计要足够好,否则视频的视觉效果会比较差。为了在帧内使用这个技术,一些标准(例如 MPEG-2)也允许帧内编码的运动向量的获取(即:既把它们当作帧间,也当作帧内)。如果块没有错误,这些运动向量就会被丢弃。

## 2. 结合时间、空间和频率的一致性

不仅仅依靠运动向量的时间一致性,我们还可以结合空间和频率的一致性。通过使用接收到的系数和同一帧中的相邻块来得到估计丢失块系数的规则,我们可以对帧内和运动向量已经损坏的帧进行错误隐藏。另外,与使用运动向量进行预测结合起来,我们可以更好地估计预测错误块。

通过将定义在块和邻近块上的平滑函数的错误减到最小,丢失块的系数可以在空间上估计出来。为了简单起见,平滑函数可以取块中邻近像素对的方差和。函数的未知量就是丢失的系数。在运动向量信息可用的情况下,预测的光滑度加到目标函数中以进行尽量化简。

502

上述这种简单的平滑方法有一个问题,它把边界也变平滑了。我们可以通过把平滑标准的阶数从线性提高到二次或者三次来改进它。这会增加边界重建和边界方向上平滑的可能性。如果考虑大量的计算开销,我们可以使用一种边界适应的平滑方法,由此,最先确定块内的边界方向,并禁止穿过边界的平滑。

## 3. 高频系数的频率平滑

为了降低计算开销,平滑可以定义得更加简单一些。虽然人类的视觉系统对低频更为敏感,但在不该出现的地方看到棋盘格图形也是让人难以接受的。当高频系数被赋予过高的值的时候,这种现象就会发生。最简单的解决方法就是把高频系数设为 0(如果它们被损坏的话)。

如果邻近块的频率是相关的,就有可能直接从频域估计出丢失的系数。对于块中每个丢失的频率系数,我们通过对四个相邻块的相同频率系数作插值来估计这些值。只有当图像是规则模式的时候,这种方法才对高频有效。遗憾的是,自然的图片通常都不是这样,因此高频系数通常直接设为 0。时间预测错误块在所有频率下的相关度甚至都更小,因此,这种方法只对帧内有效。

## 4. 运动向量丢失的估计

运动向量的丢失会使得整个预测块的解码无法完成,因此,对运动向量进行估计是很重要的。估计丢失的运动向量的最简单的方法就是把它们设为 0。当运动较少的时候,这种方法的效果不错。更好一点的方法就是检查参考宏块和邻近宏块的运动向量。假设运动也是一致的,那么可以把相应的参考帧中的宏块的运动向量作为已损坏的目标宏块的运动向量。类似的,假设持续运动的对象占用不止一个宏块,那么损坏宏块的运动向量就可以由正确接收到的邻近块的运动向量的插值得到。典型的简单插值方案是加权平均。同样,运动向量的空间估计可以与使用加权平均的参考帧估计结合使用。

### 17.3.4 前向纠错

有些数据对于正确解码是极其重要的。丢失的 DCT 系数是可以估计得到的,而且它们引起的问题在某种程度上是不可见的。然而,某些丢失或者无法准确估计的数据,比如图像编码模式、量化级别,或者视频标准协议栈的高层中的大部分数据,将会引起灾难性的解码错误。在这些情况下,我们希望能够保证“无错”传输。然而,大部分通道(特别是无线通道)是有噪声的,为了保证正确进行传输,我们必须提供足够的重传冗余(如果没有后向通道可用)。

503

前向纠错 (Forward Error Correction, FEC) 就是一种通过在位流中加入冗余数据来恢复一些随机位错误的技术。理想状态下, 通道的包错误率 (或者说位错误率) 已经估计出来, 同时加入了足够多的冗余数据来保证 FEC 恢复后的错误可能性比较低。

包错误率可以通过时间间隔来估计, 这里选取了最小可能的间隔 (目的是最小化等待时间和计算开销), 它可靠地估计了帧丢失的可能性。很自然, 当猝发帧丢失发生的时候, 估计的值可能不再合适。

帧错误也叫做疑符, 因为整个包在发生错的时候都丢失了。视频必须在一个带宽有限的通道上传输。因此, 将冗余减到最少是很重要的, 否则它会占用视频源编码可用的码率。同时, 为了使视频能够在当前的通道错误条件下保持所需的 QoS, 又需要足够的冗余。对于给定的通道环境, 都存在一个最佳的冗余可以使视频的失真降到最小。

FEC 编码通常分为两类: 块编码和卷积编码。块编码同时应用到一组数位以立刻生成冗余。卷积编码一次应用在一串位上并拥有内存来保存前面的位。接下来将简要介绍这两种 FEC 编码 [13]。

### 1. 块编码

块编码[2]取  $k$  位作为输入, 并添加上  $r=n-k$  位 FEC 数据, 共有  $n$  位长的串。这种编码称为  $(n, k)$  码。块编码的有线性和循环两种类型。所有的错误修复编码都是通过有效源串之间加入间隔来起作用的。这种间隔使用海明距离 (Hamming distance) 来测量, 定义为任意的编码串之间的最小位数, 这些串都需要改变为与第二个串一样。

要检测  $r$  个错误, 海明距离至少要为  $r$ , 否则, 损坏的串可能看上去又变成有效的了。但是这还不足以修正  $r$  个错误, 因为有效编码之间并没有足够的距离来选取更好的修正。要修正  $r$  个错误, 海明距离至少必须为  $2r$  [14, 15]。线性编码计算比较简单, 但是比循环编码的开销更高。

循环编码是通过最大次数与源位数量相等的母多项式来建立的。源位就是多项式的系数, 冗余通过与另一个多项式相乘来生成。由于求模运算实际上是平移了多项式系数, 所以这种编码是循环的。

最常用的一种循环编码方法是 Bose-Chaudhuri-Hocquenghem (BCH) 编码, 因为它能够应用于任何二进制串。BCH 的母多项式通过 GF (2) (二元 Galois Field) 给出, 它是以  $\alpha^i$  为基的多项式中次数最小的一个, 这里的  $\alpha$  是字段的一个素元 (如 2), 而  $i$  的取值是 1~2 倍的希望修正的位数。

由于使用了 Galois 字段, 因此 BCH 编码能够使用整数运算快速地编码/解码。H.261 和 H.263 使用 BCH 为每 493 个源位留出 18 个校验位。遗憾的是, 18 个校验位最多只能修正源中的两个错误。因此, 包仍然容易受到猝发位错误和单包错误的影响。

可以应用于多包的一个重要的 BCH 编码子类是 Reed-Solomon (RS) 编码。RS 编码使用了 GF ( $2^m$ ) 上的一个生成多项式, 其中  $m$  为以位为单位的包的大小。RS 编码取一组  $k$  个源包, 并输出  $n$  个包, 其中有  $r=n-k$  个冗余包。如果我们知道疑符点, 那么最多可以从  $n$  个编码后的包中恢复  $r$  个丢失的包。否则, 就像所有的 FEC 编码一样, 恢复只能应用于一半数量的包 (位的数量也是类似的), 因为在这种情况下, 错误点检测是必须的。

在 RS 编码中, 只有  $\left\lceil \frac{r}{2} \right\rceil$  个包可以恢复。幸运的是, 在包 FEC 方案中, 包在物理层上有一些头, 这些头包含一个序列号和 CRC 编码。在大多数情况下, 有错误的包会被丢弃, 并且我们能够根据丢失的序列号得知丢失的包的位置。RS 编码在存储媒介 (比如 CD-ROM) 和可能会发生猝发错误的网络多媒体传输中使用。

使用包交错也能够增加对猝发性包丢失的恢复能力。如图 17-9 所示, 为  $h$  行, 每行有  $k$  个源

视频包生成 RS 编码。然后,按照列序传输,这样,  $h$  行中每行的第一个包将首先传输,接下来传输第二个包,依此类推。如果出现猝发的包丢失,由于有足够的冗余数据,我们可以忍受多于  $r$  个的疑符。这种方案会引入额外的延迟,但是不会增加计算开销。

RS 编码可用于在基于包的网路上传输。如果出现猝发包丢失、包交错和包序列化,我们就有可能检测到接收错误的包,并使用可用的冗余来恢复它们。如果视频具有可扩展性,在基础层应用充分的 FEC 保护可以更好地利用分配的带宽,包含了将视频解码为最小 QoS 所需的运动向量和所有的头信息。可以给增强层比较少的保护或者根本不保护,只依赖于编码的恢复和错误隐藏能力。任何一种方法都可以获得最小的 QoS。

块编码的一个缺点就是,它们不能有选择性地应用于特定的位。如果它们在同一个传输包(甚至是同一组包)中发送,就难以通过更多的(例如,比 DCT 系数更多的)冗余位保护高层协议层的头。另一方面,卷积编码就能到达这种要求,这使得它们对于使用非对等保护比较有利的数据(比如视频)更加有效。虽然卷积编码对猝发包丢失不那么有效,但是对于无线通道来说,猝发包丢失并不占主导地位(并且在大多数传播模型中都不出现)。

## 2. 卷积编码

卷积 FEC 编码也定义在生成多项式上[13]。它们通过将  $k$  个消息位移入一个编码器计算得到,编码器会将它们与生成多项式进行卷积来产生  $n$  个位。这种编码的码率被定义为  $\frac{k}{n}$ 。由于编码是使用存储(移位)寄存器得到的,所以移位是必须的。其中可能会有  $k$  个以上的寄存器,在这种情况下,过去的位也会影响冗余码的生成。

在产生  $n$  个位以后,某些冗余位可以被删除以减少  $n$  的大小,并提高编码的码率。这种 FEC 方案称为码率兼容删除型卷积(RCPC)码。码率越高,位保护就越低,不过码率的开销也会变低。具有软决断的 Viterbi 算法对编码后的位流进行解码,尽管涡轮编码更为流行。

因为猝发包丢失发生的可能性不大,所以 RCPC 编码在无线(部分)网络上比块编码更有优势。RCPC 删除在校验信息生成后完成。已知对视频质量比较重要的源位,我们可以应用不同的删除量,因此错误保护的程度也有所不同。对无线模型的研究和模拟说明,如果分配的码率相同,按照位的重要性信息来使用 RCPC 应用不等量的保护会比使用 RS 编码进行视频保护得到的视频质量更好。

为简单起见,应该对视频协议中的图像层进行最高级别的保护,对更为本地化的宏块层进行级别低一些的保护,而块层中的 DCT 系数只需很少的保护或者几乎不用保护。对于可扩展的视频也可以使用这种方法进行扩展。

cdma2000 标准对任意数据类型都使用卷积编码来保护传输的位,只是在不同的传输码率下的编码率有所不同。如果未来的 3G 网络加入数据类型相关的规定,并能够识别出传输所选择的标准,它们就能够自适应地应用视频流的传输编码,并提供足够的适合当前通道状况和 QoS 要求的不等量冗余。

### 17.3.5 无线交互式多媒体的趋势

UMTS 论坛预测,到 2010 年,全世界无线多媒体通信的用户数量将会超过 10 亿,这种通信

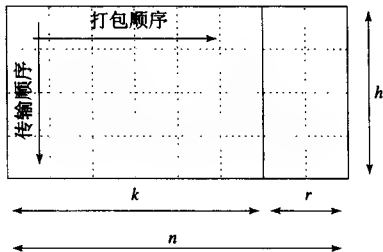


图 17-9 冗余编码的交错方案。包或者位按行存储,冗余在最后  $r$  列中生成。发送按列进行,从上到下,从左到右

量会为运营商带来上千亿美元的价值。另外, 3G 将会加速电信、计算机、多媒体内容和内容提供商的融合, 以支持功能更强大的服务。

近年来, 全球大部分蜂窝式网络都提供了 2.5G 服务。最初的 3G 服务也开始在全球范围内提供, 很多国家都已经有了商用的 cdma200 1X 服务。

506

当前与将来的一些 3G 应用有:

- 多媒体消息服务 (MMS), 一种新的消息协议, 结合传统的文本消息, 为移动电话提供集成了音频、图像和其他多媒体内容的多媒体数据。
- 移动视频电话、VoIP 和语音驱动的网络接入。
- 拥有音频流和视频流服务的移动因特网接入。
- 移动 Intranet/Extranet 接入, 提供与公司 LAN、虚拟个人网络 (VPN) 和因特网的安全接入。
- 基于移动门户, 提供随时随地访问个性化内容的定制信息服务。
- 移动多人在线游戏。
- 普适计算[6], 比如汽车的远程信息服务, 其中, 装配有 GPS 和语音识别的汽车导航系统能够与驾驶员交互来避免开车的时候阅读地图。

一直以来, 工业界都在关注 IT、娱乐和电信的融合。电信领域的一个主要分支就致力于手持无线设备——移动站 (手机)。同时, 计算机界也在关注着开发至少能够为忙碌的人们完成一些必须的重要任务的手持计算机。手持计算机可分为 Pocket PC 和 PDA。

Pocket PC 通常比较大, 装有键盘, 能够支持桌面 PC 的大部分功能和程序。PDA 完成的任务比较简单, 比如存储备忘录和电话号码。PDA 通常使用某种手写识别方式进行输入, 虽然一些 PDA 也有键盘。PDA 制造商力争支持更多的类似 PC 的功能, 同时提供无线包服务 (包括 IP 通话), 这样, PDA 既能够用于无线因特网连接, 又具有电话的功能。

和所有的可移植计算机一样, 人机交互 (HCI) 的问题比起桌面计算机更为明显。由于没有足够的放置键盘的空间, 大多数人认为语音识别将代替命令行输入。

大多数新的 PDA 产品支持图像和视频捕获、MP3 播放、e-mail 和无线协议 (如 802.11b 和蓝牙) 当连接到 GPRS 和 PCS 网络 (例如 Handspring Treo) 的时候, 一些 PDA 还可以作为手机使用。它们拥有彩色屏幕, 并能够支持网页浏览和多媒体 e-mail 消息。一些具有蓝牙功能的 PDA 通过兼容蓝牙的手机来接入移动网络。然而, 随着手机的功能越来越强大, PDA 加入 802.11b 接口卡, 蓝牙的发展空间逐渐被压缩。

当 PDA 制造商放眼未来, 他们希望不仅仅在无线网络中支持语音通信, 还希望支持多媒体 (比如视频通信)。一些 PDA 整合了具有闪光灯和变焦功能的高级数码相机 (例如 Sony 的 CLIE)。

507

视频的编码可以使用 MPEG-4 或者 H.263 来完成, PDA 也能够支持多种播放格式。

手机制造商则尝试着加入更多类似计算机的功能, 包括 PDA 所支持的基本任务、网页浏览、游戏、图像和视频捕获、e-mail 附件、流视频、视频会议等。交互式多媒体的需求稳步增长, 特别是在图像和视频通信方面。大多数手机制造商和移动服务提供商已经能以 e-mail 附件形式、视频流, 甚至视频会议的形式支持一些图像或视频通信。和短消息服务 (SMS) 类似, 新的消息协议多媒体消息服务 (MMS) 已经得到工业界的支持, 作为带宽限制下的一种过渡解决方案。新的手机已经有了彩色的显示屏和内置的数码摄像头。大多数手机使用集成的 CMOS 传感器, 有的手机甚至拥有两个。到 2004 年, 移动电话上的摄像头数量将会超过世界范围内销售出的数码相机数量。

近年来, 手机已经能够支持网页浏览和 e-mail 功能, 但是利用包服务、蓝牙和 MMS, 它们能够支持各种格式的视频流和 MP3 播放。一些手机甚至包含使用手写识别和指点杆的触摸屏, 就像大多数 PDA 一样。另外的一些手机已经小到可以穿戴起来来代替手表了。

## 17.4 进一步探索

Tanenbaum[14]对无线网络进行了比较全面的论述,而 Wesel[2]对无线通信网络进行了介绍。Viterbi[5]对扩频和 CDMA 的基础进行了严谨的分析。Wang 等[16]对视频通信的错误控制进行了深入讨论。

这一章在网站上的 Further Exploration 部分包括关于无线网络的网上资源:

- 关于无线网络和移动电话技术的一个调查。
- 关于 GSM 的报告。
- GPRS 入门。

相关的一些链接有:

- NTIA——频谱管理方面的信息。
- CDMA 开发组、IMT-2000、UMTS、cdma2000 RTT、3GPP 等的主页。
- 无线 LAN 标准。

我们也给出了一些 PDA 和现代移动电话的图片。

## 17.5 练习

1. 在 TDMA 系统的实现(比如 GSM 和 IS-136)中,以及基于 CDMA 的低一级的网络(比如 IS-95)中,仍然使用把分配的载波频谱划分为更小的通道的 FDMA 技术。为什么这是必要的?
2. 讨论 GSM/GPRS 和 WCDMA 实现可变码率传输的方法的区别。
3. 我们在图 17-1 中看到了蜂窝网络的几何布局。图中假定蜂窝是六边形,平面对称的(即在不同蜂窝上分割频谱的方案是一致的)。而且,重用因子  $K=7$ 。根据蜂窝的大小和无线电的干扰,重用因子可能不一样。如果依然采用六边形蜂窝,所有可能的重用因子都能得到一个对称的平面吗?哪些能够得到?你能推导出一个关于可能的重用因子的公式吗?
4. IS-95 的扩展增益是什么?假定所有的用户都希望用最大的码率传输,WCDMA 的 UTRA FDD 模式的扩展增益是什么?扩展增益之间的不同会带来什么影响?
5. 当一个便携式电话用户穿越蜂窝的边界时,必须由一个蜂窝转交给另一个蜂窝。硬(不当的)转交会造成通话中断。
  - (a) CDMA(直频)比 FDMA 或者跳频(FH)提供的转交性能更好,为什么?
  - (b) 对转交技术提出一种改进使得它可以更软一些。
6. 在 CDMA 蜂窝中,当一个 CDMA 的移动站穿越蜂窝的边界时候,会发生软转交。而且,蜂窝可以被划分为一些扇区,当移动站在扇区之间移动时,会发生更软的转交。
  - (a) 论述为什么要提供更软的转交。
  - (b) 至少举出两种转交之间另外的一个不同之处。

提示:在 CDMA 系统的转交过程中,移动站可以用比蜂窝内更低的能量来进行传输。
7. 本章讨论的大多数通道分配方案都是固定(一致)通道分配。可以设计一个动态通道分配方案来改进蜂窝式网络的性能。请提出一种这样的动态通道分配方案。
8. 2.5G 技术是为包交换服务而设计的。它提供了按需连接而不需要预先建立一个电路。这对于零星的数据猝发是比较有优势的。
  - (a) 为 TDMA 包服务(比如 GPRS)提出一种实现多访问控制的方法。
  - (b) 电路对于长的数据更有效。扩展你提出的方法,使得通道能通过一个只对第一个传输包的竞争过程。

提示:为你的方案加上限制。

9. H.263+和 MPEG-4 使用 RVLC, 它们的流的解码既可以是同步标记向前, 也可以向后。RVLC 提高了使用规则熵编码进行编码的码率。
- (a) 为什么这对无线通道上的传输很有好处?
- (b) 它比 FEC 更有效的必要条件是什么?
10. 为什么 RVLC 通常只用于运动向量? 如果你希望减小码率的影响, 需要做出怎样的改进?

## 17.6 参考文献

- [1] M. Rahnema, "Overview of GSM System and Protocol Architecture," *IEEE Communications Magazine*, 31(4): 92-100, 1993.
- [2] E.K. Wesel, *Wireless Multimedia Communications: Networking Video, Voice, and Data*, Reading, MA: Addison-Wesley, 1998.
- [3] F. Meeks, "The Sound of Lamarr," *Forbes*, May 14, 1990.
- [4] H. Holma and A. Toskala, eds., *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications*, New York: Wiley 2001.
- [5] A.J. Viterbi, *CDMA: Principles of Spread Spectrum Communication*, Reading, MA: Addison-Wesley, 1995.
- [6] J. Burkhardt, et al., *Pervasive Computing: Technology and Architecture of Mobile Internet Applications*, Boston, MA: Addison Wesley, 2002.
- [7] Third Generation Partnership Project 2 (3GPP2), *Video Conferencing Services — Stage 1*, 3GPP2 Specifications, S.R0022, July 2000.
- [8] Third Generation Partnership Project (3GPP), *QoS for Speech and Multimedia Codec*, 3GPP Specifications, TR-26.912, March 2000.
- [9] K. N. Ngan, C. W. Yap, and K. T. Tan, *Video Coding For Wireless Communication Systems*, New York: Marcel Dekker, 2001.
- [10] Y. Takishima, M. Wada, and H. Murakami, "Reversible Variable Length Codes," *IEEE Transactions on Communications*, 43(2-4): 158-162, 1995.
- [11] C.W. Tsai and J.L. Wu, "On Constructing the Huffman-Code-Based Reversible Variable-Length Codes," *IEEE Transactions on Communications*, 49(9): 1506-1509, 2001.
- [12] Y. Wang and Q.F. Zhu, "Error Control and Concealment for Video Communication: A Review," *Proceedings of the IEEE*, 86(5): 974-997, 1998.
- [13] A. Houghton, *Error Coding For Engineers*, Norwell MA: Kluwer Academic Publishers, 2001.
- [14] A.S. Tanenbaum, *Computer Networks*, 4th ed., Upper Saddle River NJ: Prentice Hall PTR, 2003.
- [15] W. Stallings, *Data & Computer Communications*, 6th ed., Upper Saddle River, NJ: Prentice Hall, 2000.
- [16] Y. Wang, J. Ostermann, and Y. Q. Zhang, *Video Processing and Communications*, Upper Saddle River, NJ: Prentice Hall, 2002.

## 第 18 章 数字图书馆中基于内容的检索

### 18.1 如何检索图像

图 18-1 中显示的是名画“享乐的花园”（The Garden of Delights）的一部分，它由 Hieronymus Bosch（1453-1516）创作，现收藏于西班牙马德里普拉多美术馆。这是一幅著名的绘画作品，但我们也很可能在理解作者的创作意图方面遇到麻烦。因此，如果我们希望进行图像的自动检索，我们就不难理解用机器来提取蕴含于图像中的语义将是一个更为艰巨的挑战。对于一幅图像来说，合适的注解当然应该包含“人物”这个描述符。另一方面，这幅图像是否会被网络上用于过滤含有“裸体”信息的系统——“Net nanny”（网络保姆）过滤掉呢（见参考文献[1]）？



图 18-1 我们如何准确描述一幅图像的内容（感谢西班牙马德里普拉多美术馆）

与基于文本的检索相比，我们都知道绝大多数主要的网络浏览器都有一个用于在网上进行多媒体内容检索的按钮。就 Bosch 的画作而言，利用一个基于文本的检索就有可能做得很好，但是我们不会总是遇到这种特殊的图像。我们还会对那种更具有一般性的检索感兴趣，比如说检索那些具有深蓝色的天空和桔黄色的落日的场景。通过预先计算一些存储在数据库中的关于图像的基本统计信息，我们就能够检索到具有如上特征的简单场景。

起初，数字图书馆中的检索的思想源于传统信息检索的一些原则（例如，参见参考文献[2]）。这些查询的思路得以延续下去。例如，在[3]中，我们可以使用基本的信息检索技术把图像分为室内和室外两大类。在一个图像及其说明的训练集上，把每个单词在说明文档里面出现的次数除以这些单词在一个大类的所有文档里面出现的次数。另一个类似的度量方式则用于图像内容片段的统计性描述。这两种基于信息检索的度量方式常结合起来成为更加有效的分类机制。

但是，大多数多媒体检索方案都倾向于基于多媒体自身的一些有帮助的内容，而不是依赖于那些附加在多媒体上的额外文本信息。最近，人们再次关注图像中所蕴涵的语义内容这个比较深入的问题，并重申要利用附带的文本。如果数据不仅包含从图像中的物体所提取出的统计属性，而且包含与这些图像相关联的文本，那么任何一种形态（不管是文本还是图像）都能提供另一种

形态所遗漏的一些语义内容。例如，一幅红玫瑰的图像一般来说不会具有“红色”这个关键字，尽管我们通常认为它需要我们手动添加上去。因此，图像属性和与之相关联的一些描述字将彼此消除双方所存在的二义性（参见[4]）。

在本章中，我们只关注那些使用图像的特征来从数据库中或者网络上检索图像的一些标准化程度更高的系统。常用的图像特征一般都是一些统计性的度量，比如图像的颜色直方图。考察一幅彩色的图像，比如一个圣诞老人和他的雪橇。我们使用鲜红色、褐色以及肉色这些颜色特征的组合就足以作为图像特征，从图像数据库（比如说关于办公室的圣诞派对）中检索出相似的图像。

回忆一下，颜色直方图通常是一个统计每个像素的红、绿、蓝三色值的三维数组。这种结构最大的好处就在于它不必关心图像的方向（因为我们统计的是像素的值，而不是它们的方向），也不影响到对象的遮挡。一篇关于该课题的学术论文[5]引起了人们对这种所谓图像“低层次”特征的关注。

其他常用于描述图像的特征还有颜色布局（color layout），即用类似于在棋盘的哪些格画蓝色的天空，哪些格画橘红的落日来描述图像的草图。另一种常用的特征就是纹理（texture），即有一些典型的描述方式是基于边缘图像，通过对图像进行偏微分产生边缘图像，并且同时利用图像的空间闭合性以及朝向来区分边缘。该处理方式的一种实现使用的就是边缘特征的直方图。同时，纹理布局（texture layout）也可以用于图像特征。基于这些特征所设计的搜索引擎就称为基于内容（content-based）的，即搜索是通过建立在图像的统计性内容的基础上的图像相似度量来进行的。

通常，我们希望检索出和我们当前所关心的图像（比如前面提到的圣诞老人）相似的图像。一个更加面向产业的应用有可能会用于检索邮票中的某一个特定的图案。和图像数据库检索密切相关的行业或领域包括艺术廊、博物馆、时装、室内设计、遥感、地理信息系统、气象学、商标数据库、犯罪学，以及许多其他的领域。

一种更加困难的搜索是从图像中检索出某一对象（object），我们可以称之为“基于对象的查询”（query-by-object）模型。这涉及更加完善的图像内容的编目，将是一个很困难的目标。通常，用户的检索都是基于一种所谓的关联式检索（search by association），即在第一次检索返回结果之后，用户可以根据相似性对查询结果进行逐步求精。对于我们期望得到的图片中的那些具有代表性的图像，目录检索（catalog search）将返回请求集中的一个元素，比如商标数据库中的一个或几个商标。另外，查询也可以基于一幅特定的图像，如一幅艺术作品的一小部分，这就是目标检索（target search）。

在理解和评估现有的检索系统时，我们还要考量的一个指标就是，这些系统的搜索范围是比较窄的（比如商标数据库），还是足够宽的（比如商业图片集）。

对于任何系统，我们都面临着那些旨在替代人类劳动的机器系统的基本性质。参考文献[6]的作者简洁地总结了 we 面临的一些主要的障碍，他们称之为感官差距（sensory gap）和语义差距（semantic gap）。

感官差距是指现实世界中的对象与计算机中记录的该对象的信息之间的差异。

语义差距是指从视觉数据中提取的信息以及在给定情况下用户对相同数据的理解之间的差异。

虽然图像的特征记录了它的细节，但图像本身并不是这样表述的。尽管我们能够用语言描述图像所蕴涵的信息，但是，要让机器来提取图像的语义仍然是非常困难的。

## 18.2 C-BIRD：一个实例研究

现在，我们来分析图像查询过程的一些细节问题。为了使我们的讨论更加具体，我们使用由



本书作者开发的一个图像数据库搜索引擎来说明我们的讨论（参见[7]）。这个系统叫做数字图书馆中基于内容的图像检索（Content-Based Image Retrieval from Digital library, 简称 C-BIRD），CBIR 是基于内容的图像检索的首字母缩写词。（在本章的网站上的 Further Explorer 中给出了这个搜索引擎的一个 URL。）

513

### 18.2.1 C-BIRD 的 GUI

图 18-2 展示的是 C-BIRD 系统的 GUI。利用该系统可以浏览在线图像数据库，也可以以某种方式检索，这些方式包括文本注解、颜色直方图、光源恒常性颜色直方图、颜色密度、颜色布局、纹理布局以及基于模型的查找。大多数的图像都是视频的关键帧，并且在这个系统中还集成了一个视频播放器。



图 18-2 C-BIRD 图像查找的 GUI

下面我们逐一介绍这些检索方式。其他的系统（我们将在 18.3 节中讨论）也具有相似的特性。

### 18.2.2 颜色直方图

在 C-BIRD 系统中，数据库中的图像的特性都是预先计算好的。在图像数据库检索中，最常用的特性是颜色直方图[5]，这是图像的一种全局性的特性，也就是说不是将图像分成一块一块地处理，而是平等对待图像的每个区域的一种处理方式。

颜色直方图统计每个像素的红、绿、蓝三色值。下面我们用伪代码举个例子，对一幅用 8 位来表示 R、G、B 值的图片，可以得到一个有  $256^3$  个单元的直方图。

514

```
int hist[256][256][256]; // reset to 0
//image is an appropriate struct
//with byte fields red,green,blue

for i=0..(MAX_Y-1)
  for j=0..(MAX_X-1)
  {
    R = image[i][j].red;
    G = image[i][j].green;
    B = image[i][j].blue;
    hist[R][G][B]++;
  }
```

通常, 占用这么多单元的直方图我们一般是不会采纳的。一方面因为较少的单元可以消除具有相似性的不同图像之间的差异, 另一方面我们也希望节省存储空间。

图像查找的过程就是将样本图像的特征向量(这里就是颜色直方图)与数据库中的每个图像(或者说是部分图像)的特征向量进行匹配的过程。

C-BIRD 在预处理阶段计算好每个目标图像的颜色直方图, 然后在用户检索图像的时候引用它们。直方图定义得较为粗略, 我们为每个单元分配 8 位, 其中 3 位用于红色, 3 位用于绿色, 剩下的 2 位用于蓝色。

举个例子, 图 18-3 展示的是用户选择了一幅图像, 图像中有红色的花朵。从包含 5000 幅图像的数据库中检索到 60 幅匹配的图像。大多数 CBIR 系统要么返回与查询要求最相似的几个结果, 要么返回达到设定的相似度阈值的一组对象。C-BIRD 使用的是后一种方法, 所以可能出现查询结果为 0 的情况。

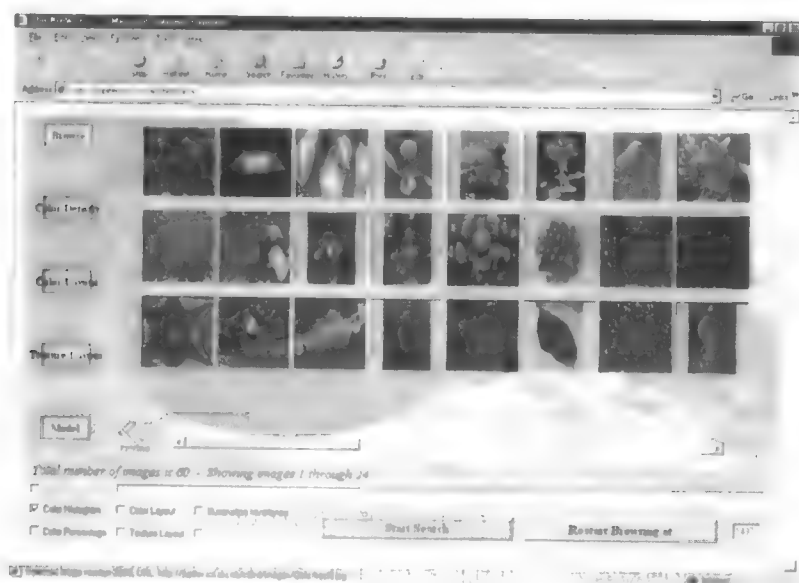


图 18-3 颜色直方图的查找结果(彩色插页中也有此图, 一些小图片来自 Corel Gallery, Corel 拥有其版权)

怎样进行匹配取决于我们采用哪种相似性度量。一种标准的颜色直方图度量方式叫做相交直方图(histogram intersection)。首先, 我们计算好数据库中每个图像  $i$  的颜色直方图  $H_i$ 。我们通常将颜色直方图看做一个三维的数组, 而机器把它看做一个很长的向量, 也就是这种度量常用的“特征向量”。

接下来, 将这些颜色直方图归一化(normalized), 使得其和(此时为 double 类型)为 1。归一化的过程非常有趣, 它有效地消除了图像的大小信息。原因如下: 如果一幅图像的分辨率是  $640 \times 480$ , 那么颜色直方图的所有项的总和将达到 307 200。但是, 如果一幅图像的分辨率是  $320 \times 240$ , 则颜色直方图的所有项的总和就只有 76 800。除以所有像素的总和就可以消除这种区别。实际上, 归一化的颜色直方图可以视为概率分布函数(probability density functions, pdfs)。颜色直方图将存储在数据库中。

假设现在我们已经选定了一幅样本图像——这幅新图像将用于和数据库中所有的可能目标

进行匹配。它的颜色直方图  $H_m$  将与数据库中所有图像的颜色直方图  $H_i$  进行求交运算, 其公式如下[5]:

$$intersection = \sum_{j=1}^n \min(H_i^j, H_m^j) \quad (18.1)$$

公式中的  $j$  代表颜色直方图的第  $j$  个单元, 而每个颜色直方图有  $n$  个单元。计算结果越接近 1, 图像匹配得越好。这种计算的速度是非常快的, 但是我们必须注意到这个求交的值对颜色的量化程度是非常敏感的。

### 18.2.3 颜色密度

图 18-4 是颜色密度的显示方案。用户可以通过一个颜色拾取器和滑动条来选择图像中某种或者某些颜色所占的比例。我们也可以选择对各种指定的颜色比例之间做“与”(AND)或者是“或”(OR)运算。这是一种较为粗略的检索手段。

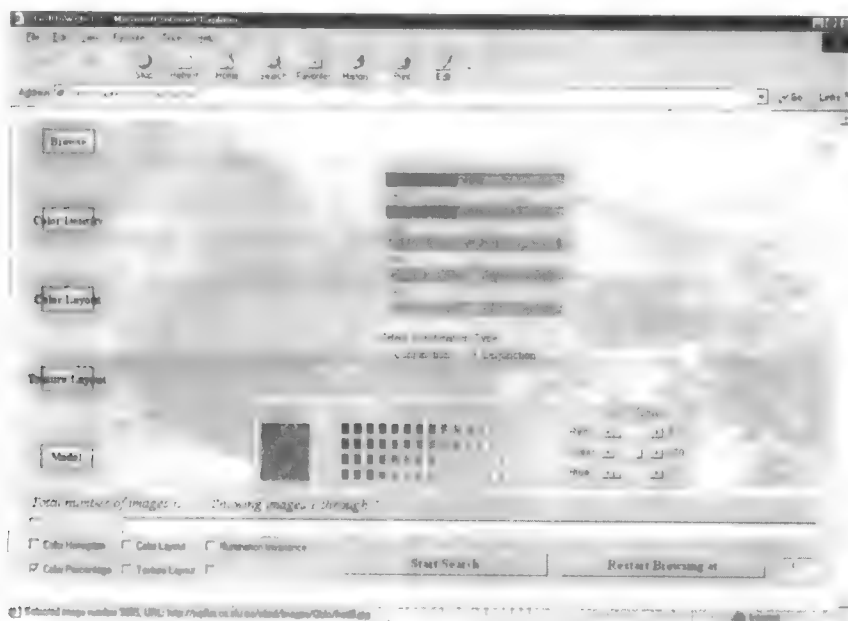


图 18-4 颜色密度的查询方案

### 18.2.4 颜色分布

用户也可以用粗略的颜色块来自定义颜色在图像中分布的草图。用户有 4 种可供选择的栅格大小:  $1 \times 1$ 、 $2 \times 2$ 、 $4 \times 4$  和  $8 \times 8$ 。可以指定在其中一种栅格上进行查找, 这些栅格被填充以一定的 RGB 值 (也可以不填充任何值, 表示这些栅格是不需要考虑的)。数据库中的每幅图像都需要分割成一些小的窗口, 对每种窗口尺寸均需分割一次, 也就是说每幅图像需要做 4 次这样的分割。对每个窗口计算聚类颜色直方图, 并在数据库中存储出现频率最高的 5 种颜色。每个查询方格的位置和大小同图像窗口的位置和大小相对应。图 18-5 展示了这种布局方案的使用方法。

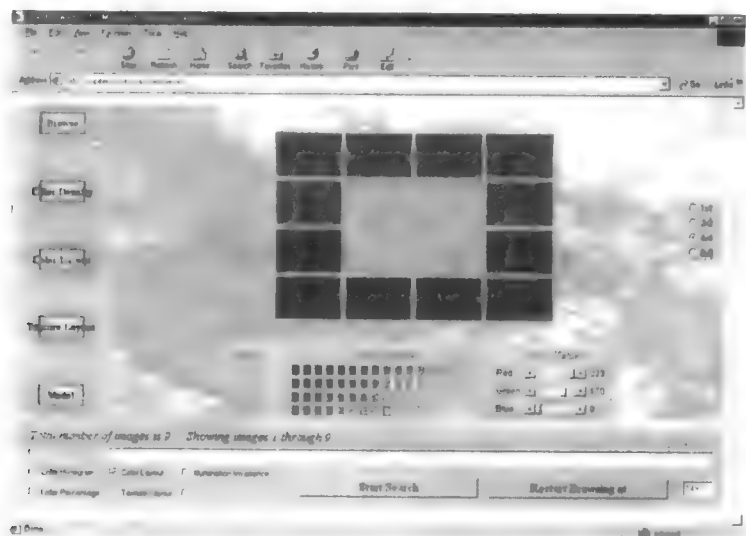


图 18-5 颜色分布的网格

### 18.2.5 纹理分布

和按颜色分布查找类似,在该方式下,用户可以通过绘制期望的纹理分布来进行查询。用户可以选择 0 密度纹理,4 个方向 ( $0^\circ$ 、 $45^\circ$ 、 $90^\circ$ 、 $135^\circ$ ) 的中密度边缘纹理及这 4 个方向的组合纹理、4 个方向 ( $0^\circ$ 、 $45^\circ$ 、 $90^\circ$ 、 $135^\circ$ ) 的高密度纹理及这 4 个方向的组合纹理。纹理匹配根据纹理的方向和密度对纹理分类,并计算分类后的纹理与用户选择的纹理分布的相关性。图 18-6 展示了这种方案的使用方法。

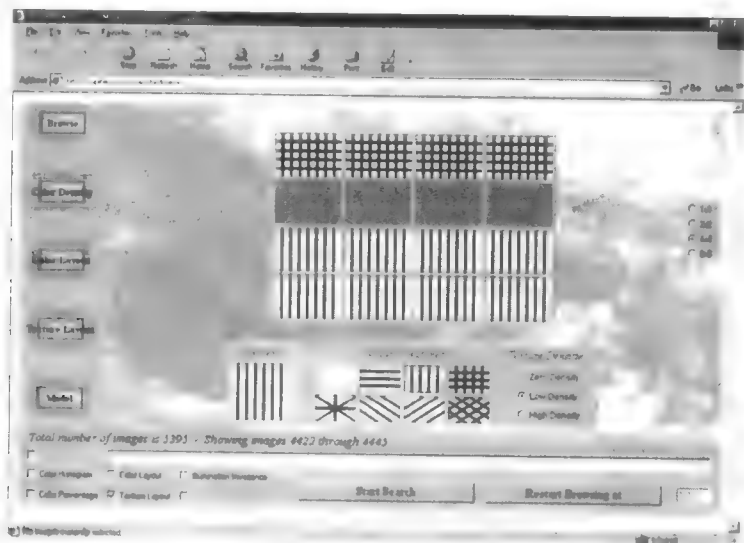


图 18-6 纹理分布网格

#### 纹理分析的细节问题

我们需要仔细考查图像检索中基于纹理的内容分析的一些细节问题。这些细节涉及系统在实际运行中必须采纳的一些技术。

首先,我们需要建立一个纹理直方图。用于理解纹理含义的常用索引结构是 Tamura 索引[8]。人类感知方面的研究表明,人们在识别纹理的时候,重复性、方向性以及间隔尺寸是最为明显的区分因子[9]。因此,我们的纹理直方图是建立在方向(directionality)  $\phi$  和边间隔(edge separation)  $\xi$  基础上的二维纹理直方图。这里的边间隔  $\xi$  和重复性有着密切的关系。 $\phi$  表示边的朝向,  $\xi$  表示平行的纹理边之间的距离。

为了提取边缘图像,首先需要通过下式将图像转化为用亮度  $Y$  表示:  $Y=0.299R+0.587G+0.114B$ 。然后对这个用  $Y$  值表示的图像利用 Sobel 抽边算子(Sobel edge operator) [10]进行抽边操作,抽边过程就是将下面的  $3 \times 3$  权系数矩阵(convolution mask)在整幅图上移动并作卷积。

$$d_x: \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \quad d_y: \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array} \quad (18.2)$$

如果我们对每个像素都按照上面的矩阵加权求和,我们都得到了我们所需要的结果的一个近似值。

边缘大小  $D$  和边缘斜率  $\phi$  定义如下:

$$D = \sqrt{d_x^2 + d_y^2}, \quad \phi = \arctan \frac{d_y}{d_x} \quad (18.3)$$

接下来,我们将通过删除所有非最大值的边来压缩边缘图像。如果一个像素  $j$  具有边缘斜率  $\phi_j$  和边缘大小  $D_j$ , 它沿  $\phi_j$  方向上的邻近像素为  $i$ , 如果  $\phi_j \approx \phi_i$  并且  $D_j > D_i$ , 那么像素  $j$  的值就被置为 0。

为了得到一幅二进制的边缘图像,我们只需要将  $D$  值大于一定阈值的像素置 1, 而将其余像素置 0 即可。

对于边缘间隔  $\xi$  采用如下的处理手段,我们对每个像素  $i$ , 沿着它的边缘斜率  $\phi_i$  求与之最近的像素  $j$  之间的距离, 像素  $j$  的边缘斜率必须在  $15^\circ$  的误差范围内满足  $\phi_j \approx \phi_i$ 。如果找不到这样的像素  $j$ , 那么我们就说这个边缘距离为无穷大。

在建立好边缘方向和边缘间隔的映射之后, C-BIRD 就构建出一个有关  $\xi$  和  $\phi$  的 2 维纹理直方图。初始化时直方图大小为  $193 \times 180$ , 其中  $\xi=193$  这个值作为表示间隔无穷大的保留值(等价于所有  $\xi > 192$  的值)。直方图的大小在两个维度上都减少为原来的  $1/3$ , 最后的大小为  $65 \times 60$ , 在这个过程中,相连接的项被求和。

接下来通过将每个像素值替换为该像素与其相邻像素的加权和来使直方图更加“平滑”, 最后的结果是得到大小为  $7 \times 8$  的图像, 这个时候我们将值 7 视为表示无穷大的保留值。在这个阶段, 纹理直方图也需要进行归一化, 归一化过程是通过将直方图除以该图像分割中像素数目而实现的。

### 18.2.6 按光源恒常性查找

光源照明度变化能够极大地改变照相机里 RGB 感应器测量到的颜色值。比如, 在强光下的粉红色在弱光下会变成紫色。

为了处理查询图像到数据库中相应图像的光照变化的情况, 首先须将每幅图像的每个颜色通道带宽都归一化, 然后压缩成一个 36 维的向量[11]。要避免光照变化所带来的颜色变化, 将每幅图像的 R, G, B 带宽都归一化不失为一种简单而有效的办法。接下来, 我们利用色度(chromaticity)来建立一个二维的颜色直方图, 这实质上是一个带宽比值  $\{R, G\}/(R+G+B)$  的集合。这里所说的色

度和视频中的色度很类似,视频中的色度只捕捉颜色信息,而不包含照明度(或称亮度)的信息。

接下来,我们得到的一幅  $128 \times 128$  的 2D 颜色直方图可以视为一幅图像,并使用基于小波变换的压缩方案将其压缩[12]。为了进一步减少特征向量中向量分量的数目,我们计算出更小的直方图的 DCT 系数,并按照 Z 字排列,通过这些变换我们就得到了只有 36 个分量的结果。

匹配在压缩的空间上进行,通过比较两个经过 DCT 压缩的具有 36 个分量的特征向量之间的距离来决定匹配的程度(这里的按光源恒常性查找方案和下面将要介绍的按对象模型查找都是 C-BIRD 系统所独有的)。图 18-7 显示了这种查询的返回结果。

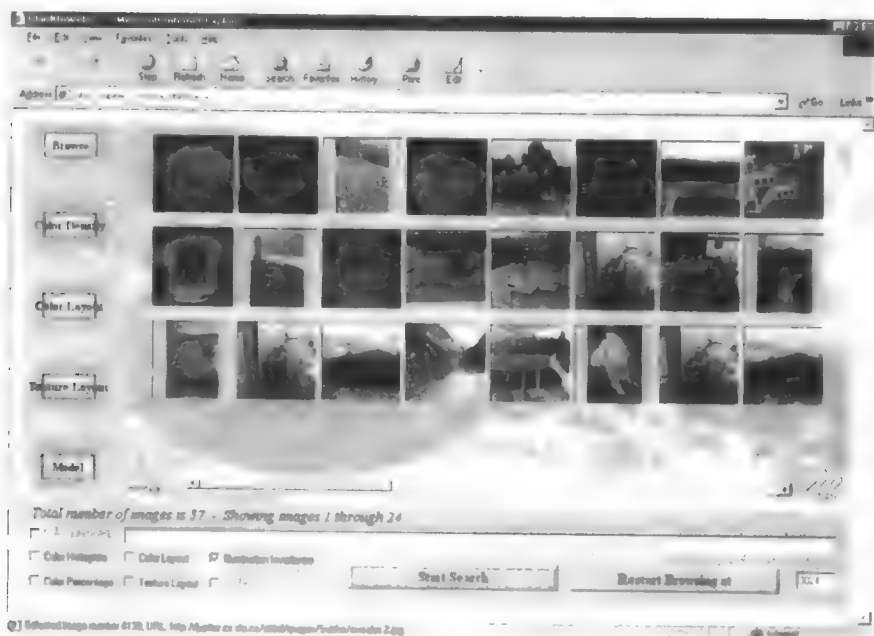


图 18-7 按光源恒常性查找(其中一些小图片来自 Corel Gallery, Corel 拥有其版权)

上面介绍的几种查找方式都可以通过选择复选框中的某一个选择框而一次性完成。返回结果是一个经过简化的图像列表,这个列表由单独使用各种查找方式所得到的结果关联而成。

### 18.2.7 按对象模型查找

C-BIRD 支持的最重要的查找方式就是基于对象模型的查找。用户可以选择一幅样本图像并在该图中选取一个特定区域来进行按对象模型查找。在不同场景条件下拍摄到的对象也能有效地进行匹配。使用这种查找方式时,用户先选择一个较小的区域,然后点击 **Model** 按钮进入 **Object Select** 模式。接着通过交互手段选择一个对象作为查询图像的一部分。下面我们将举例说明按对象检索的各组成部分。

图 18-8 展示的是对象样本的选择过程。我们可以利用一些基本的形状(如矩形、椭圆)来选择图像区域,还可以利用基于种子的扩散算法实现的魔术棒,激活的轮廓模型(图上的“snake”),或者是画刷。所有选择出的区域可以通过并、交、差等布尔运算进行组合。

一旦用户根据需要选择出了对象区域,它们可被拖至右边的面板,在该面板中显示当前选择的所有对象。可以将多个对象区域拖至选择面板,但选择面板中只有当前活动的对象能够作为检索的依据。用户可以控制扩散阈值、画刷宽度、主动寻找轮廓曲率等参数。



图 18-8 C-BIRD 界面，显示使用基本的椭圆进行对象选择（彩色插页中也有此图，本图片来自 Corel Gallery，Corel 拥有其版权）

按对象模型查找的详细实现机制在[12]中有详细介绍，我们接下来也用个系统来说明一下。图 18-9 中展示了它的算法流程框图。首先，对用户选择的图像样本进行处理以找到其中的特征（具体内容在后面介绍）。然后，我们使用在 18.2.6 中介绍过的对颜色直方图进行相交完成第一次筛选。接下来，我们需要估计对象在目标图像中的位置（通过缩放、平移或旋转）。紧接着通过对纹理直方图求交来进行验证，最后使用一个有效的通用哈夫变换（Generalized Hough Transform）来进行形状验证。

521

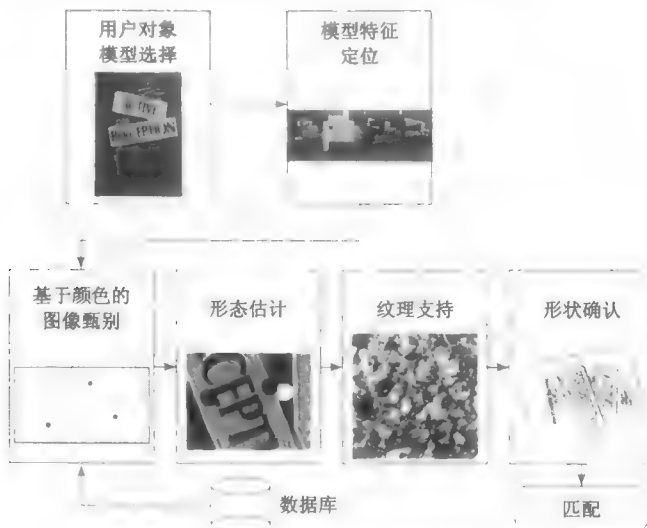


图 18-9 对象匹配步骤的方块图

图 18-10 展示了一幅用户选择的图像和一幅在数据库中存放的目标图像。显然，图像 b 应该是图像 a 检索的结果，尽管图像 b 的拍摄背景很暗。



图 18-10 模型和目标图像 (彩色插页中也有此图, Active Perception 的封面来自 Lawrence Erlbaum Associates 公司, 在此感谢!)

### 1. 特征定位的场景

前面介绍的按对象模型查找 (查找目标图像中的某个对象) 比较让人满意, 但是在检索多媒体数据方面仍然是一个实现起来比较困难的机制。在不同的光照条件下拍摄的物体, 其搜索难度就会更大。人眼的色感一致性[19]能使我们在不同的光照条件下识别出相同的物体。而对图像索引而言, 我们需要定义一个与光照变化协变的处理过程[12]。这样的话, 我们就可以补偿光照变化带来的影响。

由于按对象模型查找考查的是图像中的对象, 因此需要应用某种图像分割技术以便我们能够考察对象的每个区域, 例如, 可以用相同颜色属性来进行图像分割。尽管如此, 实际中更为有效的办法并不是对图像进行完全的分割, 而是使用一些较为粗略的区域划分, 区域之间可以有重叠 (这就是所谓的场景[7]), 然后用这些区域来表示一种粗略的特征定位。

我们在描述特征定位的处理过程的时候, 需要特别注意前面所述的场景驱动的查找方法的一些细节问题。尽管我们对光照变化比较感兴趣, 但我们也需要关注照明变化的补偿技术, 以便我们能够实现颜色协变式查找。

### 2. 特征定位与图像分割

关于图像分割 (cf. [14]): 如果  $R$  是图像分割的一个区域, 那么

- 1)  $R$  是连通的,  $R$  中的所有像素也是连通的 (8 连通或是 4 连通)。
- 2)  $R_i \cap R_j = \emptyset, i \neq j$ ; 区域之间不相交。
- 3)  $\bigcup_{i=1}^n R_i = I, I$  是整个图像, 也就是说分割是完备的。

基于图像分割的对象检索算法通过对区域匹配的度量给予一定的偏差来允许区域可以具有一定的不严格精确性。这里指的是分割时的小的非精确性 (分割时在像素级的近似而产生), 而不是指分割中的过度分割或是欠分割问题。这种技术仅适用于简化的图像, 因为其中对象的像素具有与位置无关的统计特性。

粗略的图像特征定位以近似性与简洁性为其基本思路, 它比图像分割技术更为有效, 也更容易实现。

定义: 场景  $\mathcal{L}_f$  是特征  $f$  的局部闭包。

场景  $\mathcal{L}_f$  用像素块 (简称为块) 作为定位单位, 它具有如下的描述信息:

- 1)  $\mathcal{L}_f$  包络。一个表示  $\mathcal{L}_f$  位置的块集合。
- 2) 几何参数:



质量  $M(\mathcal{L}_f)$  = 具有特征  $f$  的像素总和。

$$\text{质心 } C(\mathcal{L}_f) = \sum_{i=1}^{M(\mathcal{L}_f)} P_i / M(\mathcal{L}_f), \quad P_i \text{ 是位置。}$$

$$\text{离心率 } E(\mathcal{L}_f) = \sum_{i=1}^{M(\mathcal{L}_f)} \|P_i - C(\mathcal{L}_f)\|^2 / M(\mathcal{L}_f)。$$

522  
523

3) 场景的颜色、纹理和形状参数。比如说, 场景的色度、延展度和场景纹理直方图等。

一开始, 图像被细分为正方形的块 (例如  $8 \times 8$  或  $16 \times 16$ )。在图像分割中我们以像素为基本单位, 但在特征定位中我们是以块为基本单位。块将具有类似特征的像素归类, 如果具有特征  $f$  的像素足够多的话 (比如说大于 10%), 那么我们就说这个块具有特征  $f$ 。

在同一个位置既要很好地估算对象一级的统计信息又要表示多种特征, 就必须按块来处理。然而, 场景几何参数是按照像素来度量的, 而不是按照块来度量的。所以这里就提出了特征粒度的问题。因此, 特征定位并不仅仅是图像分割的一个简化版本。

在进行特征定位的处理之后, 下面这些断言有可能为真:

1)  $\exists f, \mathcal{L}_f$  不是连通的。

2)  $\exists f \exists g: \mathcal{L}_f \cap \mathcal{L}_g \neq \emptyset, f \neq g$ ; 场景可能相交。

3)  $\bigcup_f \mathcal{L}_f \neq I$ , 非完备性, 并不是所有的像素都被表示。

图 18-11 展示了一幅有两个红色特征场景和一个蓝色特征场景的简图。连线表示图中的块属于哪个场景包络。在图中, 我们可以很清楚地看到场景不必要是连通、不相交或者是完备的, 而颜色仍然可以被定位。

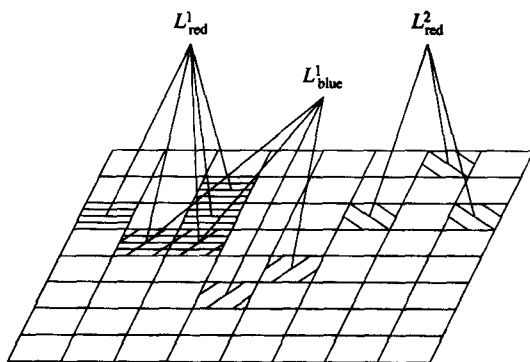


图 18-11 特征定位的场景

### 3. 块分类

在场景生成以前, 所有的块 (tile) 都需要根据某种特征进行归类, 比如红色的块, 或者红色和蓝色的块。因为在 CBIR 中, 颜色是最常用的信息, 而且平移、旋转、缩放等变换不会改变颜色, 所以我们将从颜色定位开始介绍。其他的特征 (纹理、形状、运动轨迹等) 都可以类似地定位。

### 4. 主颜色增强

为了进行颜色定位, 我们要先通过恢复图像获取过程中消除的一些颜色信息来去除噪声信号及瑕疵干扰。这里的处理是把图像从 RGB 彩色空间转换到色度-亮度颜色空间。假设像素的颜色是  $(R, G, B)$ , 我们定义

$$I = R + G + B, \quad r = R/I, \quad g = G/I \quad (18.4)$$

这里我们就把亮度  $I$  和色度  $(r, g)$  分隔开来。显然, 我们也可以使用别的具有光源恒常性的颜色模型, 如 18.2.6 节所示。

在为特征块分类之前, 我们需要把图像的像素按照具有主颜色 (dominant color) 或是过渡色 (transitional color) 来进行分类。通过考查像素和其相邻像素之间的关系来把像素划归为主像素类和过渡像素类。

定义: 主颜色是与周围像素的颜色变化无关的像素颜色, 而过渡色则与其周围像素的颜色变化相关。

524

如果一个像素在一个阈值范围内没有足够多的具有相似颜色的相邻像素，我们就可以把它视为噪声，并将其归类到过渡像素类里。我们可以使用一个  $5 \times 5$  的均值滤波器来平滑主像素类的像素，从而增强主颜色的一致性，但仅仅平滑具有相似颜色的主像素类的像素却不能增强这种一致性。图 18-12 说明，主颜色增强可以使得图 18-10 的目标图像更加清晰。

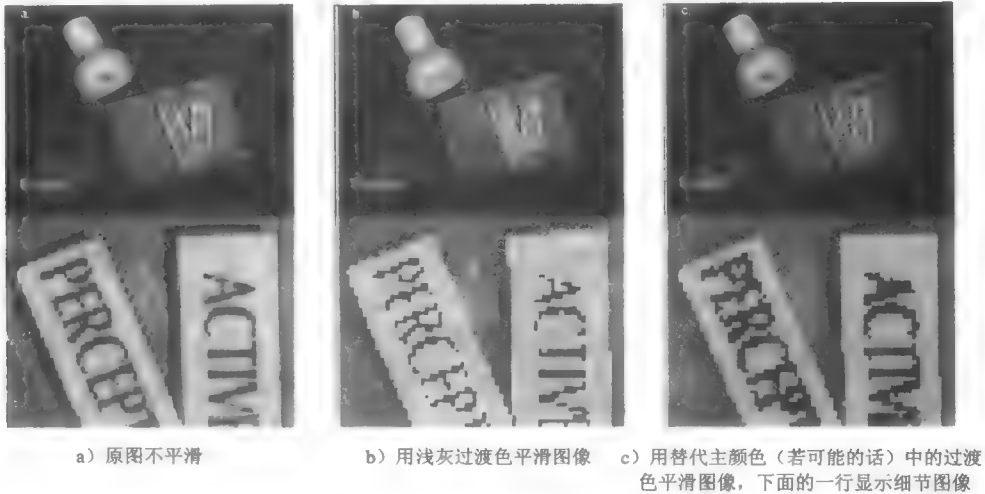


图 18-12 用主颜色进行平滑

### 5. 块特征列表

每个块具有一个块特征列表 (tile feature list)，记录这个块的所有颜色特性以及几何统计特性。在第一阶段的处理过程中，我们把主像素类添加到块特征列表中。每添加一个像素，如果它的颜色在色度-亮度颜色阈值里和列表中的某一个特征很接近，那么这个特征的颜色、几何统计信息将被更新；否则，我们要在列表中添加新的颜色特征。这个特征列表被称为主特征列表 (dominant feature list)。

在第二阶段的处理过程中，所有的过渡色都被添加到主特征列表中，在添加过程中不会修改列表中的颜色特征，而只更新其几何统计特性。为了确定当前过渡类像素应该合并到主特征列表中的哪个节点，我们要考查这个过渡像素的相邻像素，以确定它们最接近的颜色。如果相对应的颜色在主特征列表中不存在，那么我们就需要额外创建一个过渡特征列表并将这个过渡颜色添加进去。

如果一个过渡类像素  $tp$  的颜色是  $(r, g, I)$ ，那么它对应的主颜色  $(r_i, g_i, I_i)$  应该满足下式：

$$\min_{i=1}^{nc} \left\| \begin{pmatrix} r \\ g \end{pmatrix} - \begin{pmatrix} r_i \\ g_i \end{pmatrix} \right\| / F(r_i, g_i, I_i) \quad (18.5)$$

参数  $nc$  是  $tp$  的相邻像素 (邻域) 中颜色差异很大的像素个数。将相似的颜色取平均生成平均色  $(r_i, g_i, I_i)$ 。  $F(r_i, g_i, I_i)$  是第  $i$  种平均色的频度，也就是我们对多少种相近的颜色取平均来得到这个平均色。使得上式值最小的颜色将是一种最好的折衷：我们既考虑到了根据颜色相似性  $tp$  应该对应的主颜色，又考虑到了其邻域中相似颜色的种数。在我们的系统实现中，我们取这个邻域大小为  $5 \times 5$ 。

当所有的像素都添加到块中后，我们将合并主特征列表和过渡特征列表。如果过渡特征列表中的节点和主特征列表中的某个节点的颜色很接近，那么过渡特征列表中的节点将被合并到主特

征列表中的节点, 并且更新其几何统计信息。否则, 这两个节点都会保存到最终的列表中。

## 6. 场景生成

场景通过一个动态  $4 \times 4$  的叠加金字塔过程来生成[15]。每一层的父节点通过公平竞争以包容其子节点。图像块是这个金字塔最底层的子节点, 当这种竞争过程一直延续到金字塔的顶层时就会生成整个图像的场景。金字塔顶层的节点关联了一个颜色特征列表, 其中有块的集合(称之为包络)及一些几何统计特性[12]。

每个层次的竞争过程使用一个  $2 \times 2$  的非叠加链接结构来初始化, 也就是将 4 个子节点链接到一个父节点。这个初始化的过程 LocalesInit 如下所示:

525  
}  
526

过程 18-1 LocalesInit // 链接初始化的伪码

```
BEGIN
  令  $c[n_x][n_y]$  为子节点的二维数组
  令  $p[n_x/2][n_y/2]$  为父节点的二维数组
  For 每个子节点  $c[i][j]$  do
     $cn = c[i][j]$ ,  $pn = p[i/2][j/2]$ 
    For  $cn$  特征列表中的每个节点  $cn_p$  do
      在  $pn$  特征列表中找一个具有相似颜色的节点  $pn_q$ 
      If 合并后的  $cn_p$  和  $pn_q$  的离心率  $E < \tau$  then
        合并  $cn_p$  和  $pn_q$ 
      If  $pn_q$  不存在或者  $E \geq \tau$  then
        将  $cn_p$  加到  $pn$  特征列表的开头
  END
```

初始化之后, 我们就可以开始前面所谓的竞争过程。由于使用的是  $4 \times 4$  的叠加金字塔结构, 所以将会有 4 个父节点竞争与某一个子节点的链接关系。这个过程可以用 Envelope Growing 伪码描述如下:

过程 18-2 EnvelopeGrowing // 场景生成的伪码描述

```
BEGIN
  令  $c[n_x][n_y]$  为子节点的二维数组
  令  $p[n_x/2][n_y/2]$  为父节点的二维数组
  Repeat until 父子节点的链接关系不再变化
    For 每个子节点  $c[i][j]$  do
       $cn = c[i][j]$ 
       $pn \in p \left[ \left\lceil \frac{i+1}{2} \right\rceil \right] \left[ \left\lceil \frac{j+1}{2} \right\rceil \right]$ 
      For 特征列表  $cn$  中的每个节点  $cn_p$  do
        在特征列表  $pn$  中寻找具有相似颜色并使距离  $\|C(cn_p) - C(pn_q)\|$  具有最小值的节点  $pn_q$ 
      If 合并后的  $cn_p$  和  $pn_q$  的离心率  $E < \tau$  then
        删除  $cn_p$  与原来父节点的链接, 并将这个节点链接到父节点  $pn_q$ 
        更新相关的几何统计特性
      在父节点  $p$  的特征列表中删去空节点
    在金字塔中逐层向上重复如上过程
  END
```

在上述过程之后,一些小质量的场景将被删除。因为小场景一般而言不够准确,它们更可能是对象的一个微不足道的部分甚至可能是噪声干扰。为了提高检索效率,所有的场景将按照其质量降序排序。

下式为颜色更新方程,它表示的是第  $k+1$  次迭代的过程,其中父场景为  $j$ ,子场景为  $i$ :

$$(r_j^{(k+1)}, g_j^{(k+1)}, I_j^{(k+1)})^T = \frac{(r_j^{(k)}, g_j^{(k)}, I_j^{(k)})^T M_j^{(k)} + (r_i^{(k)}, g_i^{(k)}, I_i^{(k)})^T M_i^{(k)}}{M_j^{(k)} + M_i^{(k)}} \quad (18.6)$$

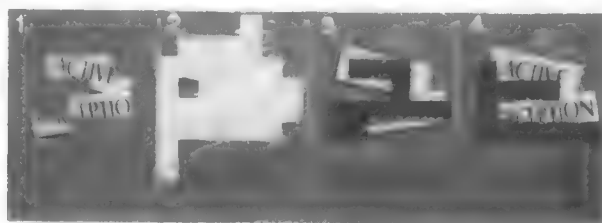
下面是几何统计特性的更新方程式:

$$M_j^{(k+1)} = M_j^{(k)} + M_i^{(k)} \quad (18.7)$$

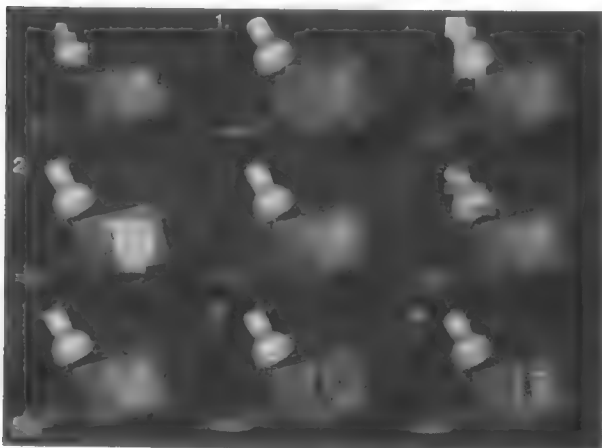
$$C_j^{(k+1)} = \frac{C_j^{(k)} M_j^{(k)} + C_i^{(k)} M_i^{(k)}}{M_j^{(k+1)}} \quad (18.8)$$

$$E_j^{(k+1)} = \frac{(E_j^{(k)} + C_{x,j}^{(k)^2} + C_{y,j}^{(k)^2}) M_j^{(k)} + (E_i^{(k)} + C_{x,i}^{(k)^2} + C_{y,i}^{(k)^2}) M_i^{(k)}}{M_j^{(k+1)}} - C_{x,j}^{(k+1)^2} - C_{y,j}^{(k+1)^2} \quad (18.9)$$

图 18-13 中显示了样本和目标图像的颜色场景。



a) 模型图像的颜色场景



b) 数据库图像的颜色场景

图 18-13 颜色场景 (彩色插页中也有此图)

## 7. 纹理分析

每个场景均与一个基于场景的纹理直方图有关,纹理直方图在 18.2.5 节中进行过介绍。在生

成边缘图像的时候,与场景相关的阈值就比较重要了。这个阈值范围通过计算场景边缘纹理直方图的量级来确定。我们通过一个高斯滤波器对纹理直方图进行平滑,再二次采样为 $8 \times 7$ 的大小,最后对其归一化。

基于场景的纹理与那些全局的纹理相比,是一种更为有效的度量方式,因为与场景相关的阈值可以进行自适应的调整。图 18-14 比较了基于场景的纹理检测与基于全局范围的纹理检测(在 18.2.5 节中讨论过)。通过对比图中台灯和香蕉的边缘图像,我们可以看出基于全局范围的纹理检测丢失了不少的边缘点,而使用基于场景的纹理检测则保留了大部分边缘点。为了绘制出基于场景的边缘图像,我们需要把所有场景生成的边缘像素都画出来。



a) 使用全局阈值的数据库图像的边缘图      b) 使用基于场景的阈值的数据库图像的边缘图

图 18-14 全局与基于场景的阈值

## 8. 对象建模与对象匹配

C-BIRD 的对象模型包含一系列已定位的特征。如前所述,它们提供了最近一次匹配的丰富的统计信息。它们间的几何关系(比如说场景的空间排列)也被提取出来。一种最好的表示方式是把各个场景的质心组合成一个向量来表示。

按对象查找可以通过平移、缩放、旋转以及灯光照明变换(更多细节请参见[12])来恢复 2D 的对象。C-BIRD 也支持组合式的查找,其中按对象查找可以和其他较简单的查找方式结合使用。此时,我们需要根据减少查找时间的原则来决定查找方式的执行顺序。我们知道,按对象查找是最为复杂的方式,所以我们最后执行这种方式的查找,而且它只是根据前面查找方式返回的结果进行进一步的检索。这样就可以大大节省时间。

用户选择出的图像对象被传送到服务器进行场景匹配。由于用户提交的对象模型的场景定位信息被认为是目标图像中应该具有的定位信息,所以我们需要在目标图像的场景和对象模型的场景实现一一对应。这个对应关系可称为分配(assignment)。

场景分配通过一系列的筛选检测来验证对象的匹配性。我们可以按照复杂性与依赖性递增的顺序来决定筛选检测的次序。图 18-9 展示了对对象匹配过程的执行步骤:a)对象模型选择和模型的特征定位,b)基于颜色的筛选检测,c)形态估计,d)纹理支持,e)形状验证。

对象匹配度量值  $Q$  的定义公式如下:

$$Q = n \sum_{i=1}^m w_i Q_i \quad (18.10)$$

其中  $n$  是需要分配的场景数目, $m$  是筛选检测的种数, $Q_i$  是在筛选检测  $i$  中得到的适应值, $w_i$  是反映每个筛选检测的适应值重要程度的加权系数。 $w_i$  可以是任意的,其总和不必为 1。而  $Q_i$  值要在  $[0,1]$  区间内归一化,这样相互之间的数字才具有可比性。

从统计的角度来看,具有较大质量(有较多的像素)的场景产生错误定位的机率较小。在大

的场景中,特征能被很好地定义,小错误也在平均水平之下,所以我们在处理这些质量较大的场景的时候能够处理得非常好。类似地,我们分配较多的对象模型场景比分配较少的场景要好,因为更多的场景可以累积更大的场景质量,从而使平均错误率降低。

我们首先分配尽可能多的场景,然后计算其对象匹配度量值并在一定的阈值范围内检查错误情况。场景可以在分配的过程中被移除或者是被替换,直到我们找到合适的匹配为止。这时得到的匹配可能就是最佳匹配,所以我们没有必要去检查别的分配。在这种情况下,我们不必检查所有可能的场景分配。

在最坏的情况下,当目标模型没有出现在检索图像中时,我们就必须检测所有可能的分配后才能认定没有匹配的图像。数据库中的图像场景以及对象模型中的场景都按照质量降序排列。

528

530

### 9. 匹配步骤

用于场景分配和验证的筛选检测步骤如下:

- 基于颜色的筛选检测 (步骤 b)
  - 按光源恒常性筛选
  - 色度投票
  - 弹性相关性
- 图像对象的形态估计 (步骤 c)
- 纹理支持 (步骤 d)
- 形状验证 (步骤 e)
- 光照变化恢复

由于光照条件很容易发生变化,因此模型与目标之间可能存在颜色的差异,在这里我们就必须采纳颜色协变匹配的思想。光线变化的对角线模型指出,红色通道通过一个整体的比例因子来响应光线的变化,绿色通道和蓝色通道也一样,只是这两者有不同的比例因子[11]。

由于每一个从模型场景到目标场景的分配都蕴涵了一种对角式的光线偏移,所以我们需要通过场景投票来决定正确的光线变化。在投票空间的一个单位里如果出现多次投票,就意味着此处是光线变化的一个峰值。利用色度投票方案,我们将所有的图像场景和模型场景匹配起来为光线变化的值投票,并将其存入一个表决矩阵中。

我们可以利用弹性相关性来估计色度偏移参数,进而估计这种从模型场景到图像场景的分配过程的可行性。这个过程通过计算正确分配的概率来返回一些可能的分配结果。在这里有一个色度偏移参数的候选集,每个候选参数都用于计算弹性相关性度量值。如果这个度量值足够高(例如高于 80%),那么弹性相关性检测过程就返回可能的分配结果,并通过形态估计、纹理支持以及形状验证来进行对象匹配测试。

图 18-15 展示了模型色度空间  $\Omega\{r', g'\}$  中弹性相关性的处理过程。模型图像有三个颜色场景  $A'$ 、 $B'$ 、 $C'$ 。图像的颜色场景  $A$ 、 $B$ 、 $C$ 、 $D$ 、 $E$ 、 $F$  的光照度都被修正到模型的光照度。虽然场景 ( $A'$ 、 $B'$ 、 $C'$ ) 和 ( $A$ 、 $B$ 、 $C$ ) 的位置并非精确一致,但是我们仍然将之视为相互匹配的实体。由于我们使用的是弹性相关性检测而不是严格的模板匹配(或相关性)方法,这就使得我们的目标场景节点可以出现在模型节点的邻域中,而不是严格的空位置对等。

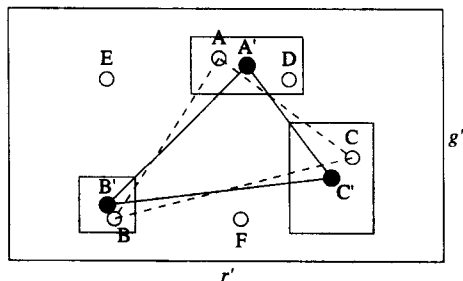


图 18-15  $\Omega\{r', g'\}$  中的弹性相关性

形态估计(步骤c)利用场景间的几何相关性来确定形态参数,因此它要在一个可行的场景分配上进行。场景的空间关系通过它们质心之间的关系来表示。更重要的是,我们仅需要2个分配的场景就能够得到足够的几何信息来恢复刚体的2D位移模型(利用4个参数,即 $x$ ,  $y$ 方向的平移量、旋转量 $R$ 和缩放参数 $s$ [12])。

形态估计的结果既是分配的最佳形态参数又是最小化的目标值,这个值指示了在刚体位移模型的意义下场景分配的匹配适合程度。如果错误率在一定的阈值范围内,我们就可以接受这样的形态估计。

支持纹理的筛选检测过程使用一种改进的直方图求交技术——将分配了的场景的纹理直方图求交。如果相交的度量值大于一定的阈值,就通过了纹理匹配。

最后的匹配验证过程是通过利霍氏变换(Generalized Hough Transform, GHT) [16]进行形状的验证(步骤e)。GHT在噪声干扰和遮挡方面具有较好的健壮性[17]。对可能的旋转、缩放、平移参数进行完全的GHT搜索将会产生非常大的开销,而且结果也未必准确。完全的搜索对大型数据库而言是不适用的。

尽管如此,在执行形态估计之后,我们已经知道了形态参数,就可以把这些参数作为模型的参考点在数据库图像中搜索具有类似参考点的图像。这样,GHT搜索就简化为仅对参考点附近一个小的邻域范围内可能匹配数的确认过程。GHT匹配方法仅需几秒钟即可完成一般的搜索。由于在计算边的斜率的过程中引入的误差在参考点处的误差值最小,所以我们通常取参考点为模型的中心。

一旦通过了形状验证,图像就会作为一个匹配结果返回给用户,如果匹配度量值 $Q$ 足够大,将同时返回 $Q$ 值。在获取了数据库中所有图像的匹配度量值 $Q_i$ 以后, $Q_i$ 值将按照值的大小降序排列。如果需要的话,可以把匹配数限制为 $Q$ 值最大的 $k$ 个匹配。在返回正确的匹配结果后,我们需要进行光照变换恢复的过程。

图18-16a显示了GHT从图18-16b所示的数据库图像中搜索红皮书的投票结果。图中暗的地方指示GHT过程在此处接收到的投票数,进而指示对象在图像的这个位置出现的可能性。18-16b展示的是为这本书重建的边缘图像。由于已经知道模型的边缘图像及其对象的定位、朝向以及缩放情况,所以这个重建的过程是自动完成的。

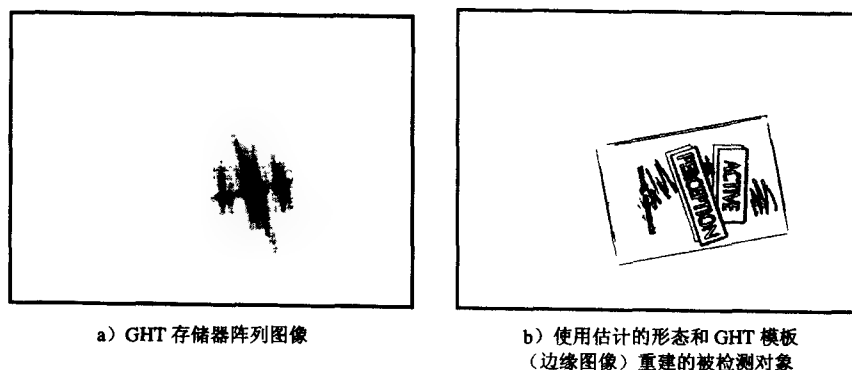


图 18-16 使用 GHT 进行形状验证

图 18-17 展示了在 C-BIRD 系统中搜索红皮书的一些结果。

由于 C-BIRD 是一个实验系统,它证明了按照对象模型进行查找的方式是可行的。



a) 仅用形态估计的查找结果



b) 使用形态估计和纹理支持的查找结果



c) 使用 GHT 形状验证的查找结果

图 18-17 按红皮书模型依据光照变化得到的查找结果  
(一些小图片来自 Corel Gallery, Corel 拥有其版权)

## 10. 视频场景

**定义:** 视频场景是指在视频中一组在时空上具有相似特征的图像特征场景的序列。

和图像中的场景一样, 视频场景也具有颜色、纹理和几何属性。此外, 视频场景还能够捕获运动参量, 比如运动轨迹和速度; 它也能够捕获时间信息, 比如视频场景的存在期以及当前视频场景与其他视频场景间的时间关系等。

由于视频是按照小的时间片来播放的, 我们就有了更为简单的处理方式, 我们可以根据前面已知的视频帧得到当前帧的场景信息, 而不必对每一帧都从头开始计算[18]。

图 18-18 展示了在加快场景生成速度后, 在帧内生成的场景和在帧间通过预测、精化得到的场景之间有一些小小的差异。

这里不对视频场景的生成做进一步的讨论。我们只提出这样一个论断: 帧间的场景生成算法通常要比帧内的场景生成算法快。此外, 视频场景提供了对实时视频对象进行分割和跟踪的有效途径[18]。



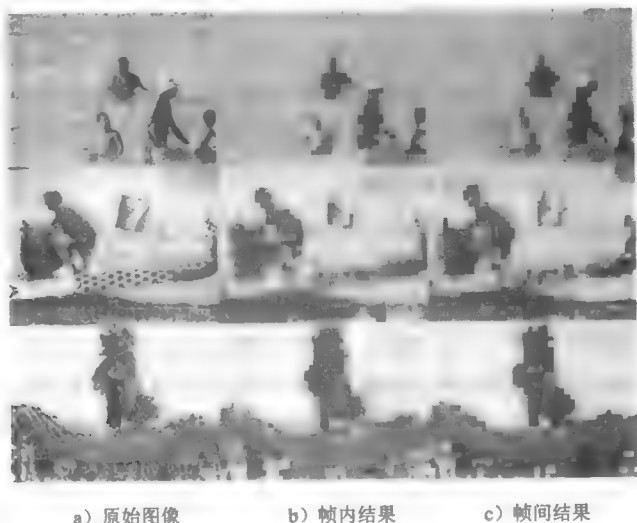


图 18-18 帧内和帧间视频场景算法的结果

### 18.3 当前图像查找系统概览

在本节,我们将对当前其他一些图像搜索引擎进行介绍,并且给出每一个图像搜索引擎的相关 URL 以供读者进一步研究(更多的 URL 和相关资料请参见本节网站第 11 章的 Further Explorer 部分)。下面的介绍并不是一个完整的概览。其中大部分搜索引擎是实验性质的成果,但是它们都具有一些比较有趣的特性。有一些系统的查询特征和前面介绍的 C-BIRD 很不相同。

#### 18.3.1 QBIC

基于图像内容的查询(QBIC)是由 IBM 位于 San Jose 的 Almaden 研究中心的研究员 Niblack 及其同事共同开发的[19]。它是现在最著名的搜索引擎。关于该引擎的更多信息可以参考 [www.qbic.almaden.ibm.com](http://www.qbic.almaden.ibm.com)。

QBIC 一个比较有趣的特征是它对颜色直方图间差异的度量标准。与式(18.1)中介绍的简单直方图求交不同,它的度量标准考虑到了相似的颜色(比如红色和橘红色)在颜色直方图上求交不应该具有 0 值。这里,它采用了颜色-距离矩阵  $A$ , 其元素为:

$$a_{ij} = (1 - d_{ij} / d_{\max}) \quad (18.11)$$

其中,  $d_{ij}$  是三维颜色距离(使用欧式距离,或者是其他一些类似的距离,如绝对值之和)。

然后,直方图差值  $D^2$  则按下式定义[20]:

$$D^2 = z^T A z \quad (18.12)$$

向量  $z$  是直方图差值向量(对向量化的直方图而言)。例如,如果二维的色度直方图是  $16 \times 16$  的,那么其直方图差值向量  $z$  的长度就是 256。

这种度量标准最大特征就是它允许我们用平均三维颜色做简单的差值来进行第一次筛选,因为这种度量标准保证其筛选的结果一定包含所有由式(18.12)筛选出的结果,然而它更简单。

QBIC 在其第一个版本的基础上又作了进一步的开发,现在已经成为 IBM 的“数字图书馆”产品套件的基本组成部分。其目的就在于提供一个完整的媒体收集管理系统。

IBM 在 QBIC 方面的另一个较有意思研究方向是将灰度级图像纳入其检索范围[21],这是一

个难度非常大的检索工作。QBIC 还可以与其他具有仅基于颜色属性的检索手段相结合, 这些手段可以是纹理理解, 比如标题和纹题。由于纹理在某种程度上反映了图像的结构信息, 所以它对灰度级图像的检索非常有帮助。当数据集变得非常大的时候, 数据库管理的问题就变得非常突出了, 这时我们要非常小心地控制簇的大小, 并需要很好地管理一个基于树的索引结构。

### 18.3.2 加州大学圣芭芭拉分校的搜索引擎

- 亚历山大数字图书馆 (ADL) 是由加州大学圣芭芭拉分校开发的一个较为成熟的图像搜索引擎。目前, ADL 主要处理地理数据: “网上的空间数据”。用户可以同一幅地图进行交互并对地图进行缩放, 而属于那些被选择的地图区域的图像作为检索结果返回。这种处理方式缓解了我们需存储大小为几 TB 的卫星图像数据的压力。ADL 使用了多分辨率的处理手段, 这使得我们可以利用较小的图像块来进行快速浏览。多分辨率图像允许我们在图像中选择一个特定的区域, 并对其进行缩放。关于这个搜索引擎的更多信息请参见 <http://www.alexandria.ucsb.edu>。
- NETRA[22]也是亚历山大数字图书馆项目的一部分。在其第二个版本 NETRA II 中, 着重强调了基于对象或是基于区域的查找过程中的彩色图像分割技术。关于 NETRA 的更多信息请参见 <http://maya.ece.ucsb.edu/Netra/>。
- 基于理解的图像检索 (PBIR) 用于实现一个更好的学习技术和更可靠的反馈技术, 它使用一种学习算法来理解用户提交的查询要求中所包含的查询意图, 并依此得到用户所需的正确查询结果。关于它的更多信息请参见 <http://www.mmdb.ece.ucsb.edu/demo/corelacm/>。

### 18.3.3 加州大学伯克利分校的数字图书馆项目

加州大学伯克利分校的搜索引擎的资料可以参见 <http://elib.cs.berkeley.edu>。这个搜索引擎主要用于商业或者其他类型的股票照片, 它也支持基于文本的检索。该系统的实验版本试图将文本中的语义信息作为图像检索的一个线索。

### 18.3.4 Chabot

Chabot 也是由加州大学伯克利分校开发的, 这是一个开发较早的系统, 其主要目标是包含 500 000 幅多分辨率的数字图像。Chabot 使用关系数据库管理系统 POSTGRES 来管理其图像以及与图像相关的文本数据。数据库中不仅存储文本数据, 而且存储颜色直方图数据。除了颜色比例以外, 系统也支持类似于“主要是红色”的这类简单查询。这个系统的更多信息请参见 <http://http.cs.berkeley.edu/ginger/chabot.html>。

### 18.3.5 Blobworld

Blobworld 也是由加州大学伯克利分校开发的。它尝试通过分割图像来获取对象的含义。为了进行恰当的分割, 它使用最大期望算法 (EM 算法) 来得到特征空间上聚类的最大似然值。Blobworld 同时支持基于文本和基于内容的检索。该系统也有不同级别的反馈机制, 在反馈中显示提交的图像的內部表示以及查询的结果, 用户可以根据反馈的信息更好地执行算法。更多信息请参见 <http://elib.cs.berkeley.edu/photos/blobworld>。

### 18.3.6 哥伦比亚大学的图像搜索器

哥伦比亚大学的一个研究小组开发了如下的一些搜索引擎:

- 基于内容的可视化查询 (CBVQ) 是该系列的第一个系统, 属于哥伦比亚大学 ADVENT 研究计划的一部分 (ADVENT 代表全数字视频的编码、网络连接及传输)。它采用基于颜色、

纹理以及颜色成分的基于内容的图像的检索方式。更多信息请参见 <http://maya.ctr.columbia.edu:8088/cbvq>。

- **VisualSEEk** 是一个彩色图像检索系统。其检索手段有基于颜色布局的检索，基于图像实例的检索（例如提供一幅种子图像的 URL），或者是根据一幅先前匹配的图像实例进行检索。VisualSEEk 支持基于可视化特征的空间关系的查询。更多信息请参见 <http://www.ctr.columbia.edu/visualeek>。
- **SaFe** 是一个集成的图像空间和特征查询系统。该系统通过从图像中分割出区域并比较这些区域的空间排列情况来进行查询。更多信息请参见 <http://disney.ctr.columbia.edu/safe>。
- **WebSEEk** 从网上收集图像（和文本）。这个系统着重于建立一个可供检索的目录结构。这些目录分别基于不同的主题，比如动物、建筑、艺术、天文等。相关的反馈是以小块图像或者是可动图标的形式提供的。对视频而言，一种较好的反馈形式是包含一些能够表示较小的视频序列的 GIF 动画图片。更多信息请参见 <http://www.ctr.columbia.edu/weebseek>（其中包括一个演示版本）。

### 18.3.7 Informedia

卡内基梅隆大学的研究项目 Informedia 数字视频图书馆项目现在已经有了第二代的成果，这就是 Informedia II。这个系统的核心是所谓的“视频挖掘”。该系统的开发得到了政府和一些大公司的支持。更多信息请参见 <http://informedia.cs.cmu.edu/>。

### 18.3.8 MetaSEEk

MetaSEEk 是一种元搜索引擎。它是在哥伦比亚大学的研究项目 IMKA 智能多媒体知识应用项目下开发的成果。其思想在于查询一些在线的图像搜索引擎，然后根据它们对不同种类的可视化查询的性能来对这些搜索引擎排序，最后针对某个查询请求，选择相应的搜索引擎来响应用户请求。更多信息请参见 <http://ana.ctr.columbia.edu/metaseek/>。

537

### 18.3.9 Photobook 和 FourEyes

Photobook[24]是一个早期的 CBIR 系统，它由 MIT 的媒体实验室开发。它利用三种不同的机制来进行三种不同类型的图像内容检索——外观、2D 形状和纹理。对前两种类型，系统首先构造一个特征函数空间——一个“特征图像”的集合，这个集合可以看作一组基图（eigenimage）。然后新的图像用相对于这个基础的坐标来表示。对纹理而言，一幅图像被分解为三个互相垂直的分量，这些分量被称之为 Wold 特征[25]。

在添加了相关的反馈机制后，Photobook 就成了 FourEyes[26]。这个系统不仅给每幅图像指定正或负的权重变化，而且对前面查询过的图像施加类似的查询操作。这样一来，这个系统的查询速度就相当快了。更多信息请参见 <http://vismod.www.media.mit.edu/vismod/demos/photobook>。

### 18.3.10 MARS

MARS（多媒体分析与检索系统）[27]由伊利诺伊大学香槟校区开发。其主要思想是建立一个动态的特征表示系统，这个系统可以适合于不同的用户和不同应用环境。其主要的手段就是适当的反馈机制（参见 18.4 节），反馈主要是用户指定的权重变化。更多信息请参见 [www-db.ics.uci.edu/pages/research/mars.shtml](http://www-db.ics.uci.edu/pages/research/mars.shtml)。

### 18.3.11 Virage

可视化信息检索 (Virage) [28] 系统对图像里的对象进行操作。它在进行一些预处理 (比如平滑处理、比较增强) 后建立一个图像索引。其特征向量的一些细节问题是保密的。但是我们知道, 这个系统对每个特征的计算并不是用一种方法而是用多种方法结合完成的, 最后的特征向量由每个单独计算出的特征组合而成。更多信息请参见 <http://www.virage.com/>。

### 18.3.12 VIPER

面向增强检索的可视化信息处理系统 (VIPER) 是一个实验系统。它关注用户指导的搜索范围缩小机制上。这涉及合适的反馈问题。这个系统由日内瓦大学的研究人员开发。VIPER 使用一个含有 80 000 种特征的巨大的图像特征集合, 这些特征基于不同层次的颜色、纹理等, 也基于图像在不同层次的分级分解。VIPER 遵循 GPL (General Public License) 下的开源软件分布系统 GNU (Gnu's Not Unix) 发布。更多信息请参见 <http://viper.unige.ch>。

### 18.3.13 Visual RetrievalWare

Visual RetrievalWare 是 Convera 公司开发的一种图像搜索技术。其开发技术来源于不同的政府部门用于检索标准文档的技术。它的图像版本大大增加了 Yahoo 的 Image Surfer 功能。Honeywell 得到了这项技术的许可证。Honeywell 收集了上百万的 X 光片, 并计划用此技术来对 X 光照片进行索引和检索。在这个软件中使用的特征有颜色、形状、纹理、明暗度结构、颜色结构以及长宽比。更多信息请参见 <http://vrw.convera.com:8015/cst>。

538

## 18.4 相关反馈

相关反馈是在最近的 CBIR 系统中所使用的强大的工具 (参见[27])。简单而言, 其主要思想是将用户置于一个操作过程的循环中, 在这个过程中, 检索到的图像用以进一步收敛正确的结果。通常的情形是由用户指定图片是好, 是坏, 还是无所谓, 而系统在用户的指导下更新各查询结果的权重。(另一种处理方式是将查询逼近于那些有正标记的内容[29]。一个更有趣的思想是, 通过空间的特征点的弯曲来移动每个数据点[30]。在后一种处理方法中, 特征点随着高维空间的扭曲而移动, 就像挤压凝胶时镶嵌在其中的葡萄干也随着运动一样。)

### 18.4.1 MARS

在 MARS 系统中[27], 指定给每个特征点的权重都可以根据用户的输入而更新。首先, MARS 的作者假定有很多种特征, 比如说颜色、纹理等, 这些特征可以编号为  $i=1 \sim I$ 。对于每个特征, 进一步假定我们可以用多种表示方式来表示。例如, 对颜色而言, 我们可以使用颜色直方图、颜色布局、瞬时颜色直方图、主颜色等多种表示方式。假设对于每一个特征  $i$ , 有  $j=1 \sim J_i$  种表示方式。最后, 对特征  $i$  的每一种表示方式  $j$ , 有一个具有  $k=1 \sim K_{ij}$  个分量的特征向量与之对应。这样, 最后我们得到了特征向量的分量  $r_{ijk}$ 。

每种特征  $i$  都有一个表征其重要性 (权重) 的系数  $W_i$ , 每个特征  $i$  的每种表示方式  $j$  的权重可以用  $W_{ij}$  表示, 而每种表示方式的每个分量则具有权重  $W_{ijk}$ 。权重是动态的, 在结合了进一步的反馈信息后可以对这些权重进行动态调整。

令  $F=\{f_i\}$  为特征  $f_i$  的全集, 令  $R=\{r_{ij}\}$  为给定特征  $f_i$  的所有表示方式的集合。接下来, 就当前特征  $i$  而言, 假定  $M=\{m_{ij}\}$  为集合  $R$  中各表示方式的相似度度量, 该度量用于确定两种表示方式的相似程度。而且, 对于不同的表示方式应该有不同的度量标准: 在比较特征向量时, 应该使用的度量标准是 Mahalanobis 距离, 而在比较颜色直方图的时候对直方图相交可能更为有效。假设

图像的原始数据集为  $D$ , 相关反馈算法可以用这样一个模型  $(D, F, R, M)$  来表示。

接下来我们看看在[29]中介绍的检索过程。

1) 初始化权系数为标准值:

$$\begin{aligned} W_i &= 1/I \\ W_{ij} &= 1/J_i \\ W_{ijk} &= 1/K_{ij} \end{aligned}$$

其中,  $I$  为集合  $F$  中特征的数目;  $J_i$  是特征  $f_i$  的表示方法的总数;  $K_{ij}$  是每个特征表示向量  $r_{ij}$  的长度。 539

2) 数据库中与查询要求相似的图像首先按照特征分量来定义:

$$S(r_{ij}) = m_{ij}(r_{ij}, W_{ijk}).$$

然后, 对每种表示方式的相似度量值加权求和:

$$S(f_i) = \sum_j W_{ij} S(r_{ij}).$$

3) 最后, 总的相似度量值  $S$  定义为:

$$S = \sum_i W_i S(f_i).$$

4) 与检索图像  $Q$  相比, 相似度最大的  $N$  个图像被作为检索结果返回。

5) 用户根据自己的主观看法对每个返回的检索结果标记上高度相关、相关、无所谓、不相关、高度不相关等反馈标志。

6) 根据用户的反馈信息, 更新各权重, 重复如上过程。

为了得到一个有意义的图像检索集合, 将该相似度量值归一化。

1) 由于不同的表示方式有不同的数值范围, 所以我们需要对特征进行归一化。其中, 既可以进行离线归一化(内部归一化), 也可以进行在线归一化(交互归一化)。

2) 内部归一化: 其主要思想是对特征表示向量  $r_{ij}$  的每个分量  $r_{ijk}$  进行归一化, 以便对每个分量赋与同等的重要性。对每个分量  $k$ , 对数据库中的所有  $M$  幅图像求其平均  $\mu_k$ 。然后用归一化后的结果替代原来分量的值, 从统计的角度来看, 常用的归一化方法如下式所示:

$$r_{ijk} \rightarrow \frac{r_{ijk} - \mu_k}{\sigma_k}.$$

3) 交互归一化: 在整体度量值  $S$  中, 我们对每个相似度量值  $S(r_{ij})$  赋予同等的重要性。我们在这里需要对数据库中所有图像的相似度量值  $S$  计算平均  $\mu_{ij}$  和方差  $\sigma_{ij}$ 。

4) 接下来, 对于任一个新的查询  $Q$ , 我们都需要通过下式在线地替换  $Q$  和数据库图像  $m$  的相似值:

$$S_{m,Q}(r_{ij}) \rightarrow \frac{S_{m,Q}(r_{ij}) - \mu_{ij}}{3\sigma_{ij}}.$$

最后, 权重的更新过程如下所示:

1) 评分值  $\{3, 1, 0, -1, -3\}$  对应于用户给出的从“高度相关”到“高度不相关”的反馈信息。

2) 根据用户对图像的反馈, 将权重按照下式更新:

$$W_{ij} \rightarrow W_{ij} + \text{评分值}$$

接下来, 按照下式对权重进行归一化:

540

$$W_{ij} \rightarrow \frac{W_{ij}}{\sum W_{ij}}$$

3) 将特征  $r_{ijk}$  标准方差的倒数赋给相应的分量的权重  $W_{ijk}$ :

$$W_{ijk} = \frac{1}{\sigma_{ijk}}$$

也就是说偏差越小, 权重越大。

4) 最后, 将这些权重归一化:

$$W_{ijk} \rightarrow \frac{W_{ijk}}{\sum W_{ijk}}$$

将用户置于一个循环的操作过程中一个最大的好处就是, 通过相关反馈, 用户在查询时不必给出完全精确的初始查询值。相关反馈在较底层的特征和较高层的概念之间建立一个更为精确的联系。CBIR 系统就是采用这样的处理方式改进了它的检索性能。

#### 18.4.2 iFind

微软的 iFind 系统[31]是明确地在图像检索中使用相关反馈机制的实验系统。它力图通过提取图像中的语义内容来捕捉图像低层次的特征。所有图像有相关的关键字, 基于这些关键字可建立一个语义网络来对图像进行访问, 并在其中集成了一些低层次的特征。每个关键字都有与数据库中相关图像的链接, 而每个链接都有一个权重。在每次用户相关反馈之后, 相关性的程度(也就是前面说的每个链接的权重)都会被更新。

显然, 一幅图像可以和多个关键字相关联, 每个关联具有不同的相关性程度。那么现在的问题就是这些关键字从何而来呢? 可以手动生成关键字, 也可以利用网络浏览器从与图像相关联的 ATL HTML 标记中检索出关键字。

### 18.5 结果的量化

总体而言, 我们需要一些简单的表示图像搜索引擎的性能表达式。

在信息检索理论中, 准确率 (precision) 定义为检索得到的相关文档数与检索得到的所有文档数的比值。而查全率 (recall) 是用户最终检索出的文档数目与所有相关文档数的比值。准确率和查全率常用于在图像检索系统中报告检索的性能。不过, 这些度量值也受到数据库大小、数据库中相似信息的数目等因素的影响。而且, 它们也没有考虑到模糊匹配或搜索结果排序等因素。

我们可以把这两个值用式子定义如下:

$$\begin{aligned} \text{准确率} &= \frac{\text{返回的相关图像数}}{\text{所有返回的图像}} \\ \text{查全率} &= \frac{\text{最终返回的图像数}}{\text{所有期望返回的图像}} \end{aligned} \quad (18.13)$$

541

总的来说, 我们将阈值范围放得越宽, 允许返回的图像越多, 那么准确率会越低, 但是查全率反而会越大。准确率与查全率的变化曲线称作接收端作业特征曲线 (ROC 曲线)。这个曲线在一定的参数范围内表示了检索灵敏度和特异性之间的关系。

#### 18.6 视频查询

对于各种查询, 可以利用运动作为瞬时变化图像的显著特征来实现视频检索。在这里, 我们

不准备详细讨论视频检索,但是会引用文献[32]中的相关内容(其中对于视频检索进行了很好的描述)。

简而言之,由于瞬时性是视频和图像集之间的主要区别,所以对时间分量的处理就成为在检索、浏览、搜索和视频内容检索工作中的首要任务。QBIC小组[21]主要致力于对视频自动理解继而产生情节串联图板——也就是所谓的“逆好莱坞”(inverse Hollywood)问题。在拍摄视频的时候,编剧和导演从一段对剧情发展的详细描述作为开始。而从视频理解的角度来看,我们则希望能重构出这个详细的剧情描述来作为“理解”视频的开始。

首先我们需要把视频划分成一些连续的镜头,而这些连续镜头大致包括在“记录”按钮被点击和被释放这段时间里记录的视频帧。视频转变(如渐现、渐隐、溶解、擦除等)往往在这些连续镜头之间发生,所以,对于连续镜头边界的检测不会像检测“突变”那样简单。

一般说来,我们所处理的是数字视频,如果可能的话,我们希望避免处理解压缩的MPEG文件,从而能够增大吞吐量。所以,研究者们一直研究压缩视频。一个简单方法是只部分解压缩直到能够还原DC项即可,这样就产生一个只相当于完全解压缩得到文件1/64大小的对象。由于我们必须考虑P帧、B帧以及I帧,所以即使要产生对最好的DC图像的一个较好的近似,也是一个复杂的问题。

一旦从视频中提取了DC帧,那么就可以应用许多方法来寻找连续镜头的边界。这些方法常用的典型特征有颜色、纹理和运动向量(这些概念在物理轨迹研究中也经常使用)。

我们可以对连续镜头进一步进行归类,组成一系列场景(scene)。一个场景是在时间上连续并且聚集在一起的连续镜头的集合。在所谓的电影语法(film grammar)中甚至会包含较高级的语义[34]。诸如剧情的基本元素之类的语义信息是可取得的。这些语义信息(粗略地说)包括剧情的介绍、评论、高潮以及结局等。

音频信息对于场景的聚合很重要。在一个典型的场景中,音频是没有断点的,即便该场景包含许多连续镜头。在电影制作过程中,一般的定时信息也会包括在音频中。

利用目前可用的闭路字幕信息,文本也可以成为描述连续镜头和场景最有用的方式了。但是,仅仅依赖于文本是不可靠的,因为它未必总是存在,对于遗留视频而言更是如此。

542

为了更合理、更简洁地组织和展示情节串联图板,目前已提出许多不同的方案。最直接的方法莫过于只演示一段关键帧的二维数组。但仅仅是如何建立一个好的关键帧的问题仍是业内激烈争论的问题。另一个方法就是每隔几秒就输出一帧。这种方法的问题就是不会顾及在两个较长时间的的非活动剧情之间是否有发生活剧情趋势。因此,可以使用某种聚类方法来代表一段较长的时间,类似于一个关键帧中的一个瞬时时间。

一些研究人员提出使用基于图形的方法。假设我们有一个视频包含两个谈话的人,分别是访问者和被访问者。一种恰当的方式是使用带箭头的有向图来表示一个谈话者到另一个谈话者的转变。用这种方法,我们就能够获取许多关于视频结构的信息,而且能够形成图形剪除和管理的工具库。

另外,人们还使用了“代理”来代表场景和连续镜头。一个关键帧集合的组合可能比一个简单的关键帧序列的表达能力更强,就像关键帧的大小可变一样。如果要合理地理解视频,那么我们需要对每个“快速掠过”(skimmed)视频关键帧集合的文本和声音加上注解。

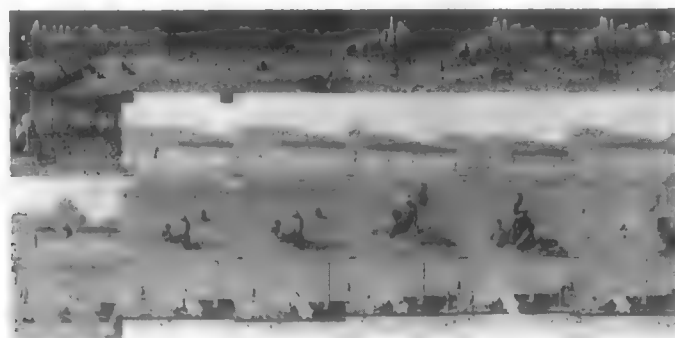
如果对于一系列帧进行特征匹配,那么帧可以结合起来形成更大的帧,这种多帧镶嵌(mosaic)的方法是很有用的。上述方法产生的更大的关键帧或许更加具有代表性。

一个更加极端的视频表示方法是选择(或创建)一个能够完全代表整个影片的帧。该帧的选择可以是基于帧中的人物、动作等。在[35]中,Dufaux提出了一个基于动作测量(利用帧间差异)、

空间活动（通过像素值分布的熵）、肤色像素和脸部检测来选择连续镜头和关键帧的算法。

如果考虑到肤色和面容，该算法将大大增加那些包括人物和肖像的关键帧的相似度，比如说利用特写镜头就是一个很好的例子。可以使用已标记的图像样例来对肤色进行“学习”，而脸部的检测可以使用神经网络来完成。

图 18-19a 展示了从一部关于海滩活动的视频（见[36]）中选出的一些关键帧。图 18-19b 中的关键帧主要是基于颜色信息选择出来的（但请注意，在截取视频时由于光照条件改变而导致了一些变化）。



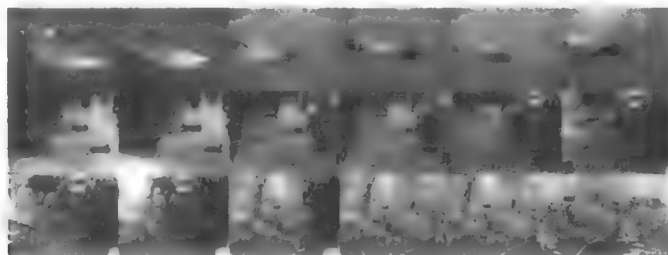
a) 一段数字视频的帧



b) 选出的关键帧

图 18-19 数字视频及其关键帧，“海滩”视频

当连续镜头之间颜色有渐变和总体的颜色又比较接近的时候，将会出现一个更加复杂的问题（如图 18-20a 所示）。图 18-20b 中的关键帧能充分说明整个视频序列的演变过程。



a) 一段数字视频的帧



b) 选出的关键帧

图 18-20 花园视频

其他的方法都试图从人这个更有意义的角度来处理视频，而不是使用较低层次的视觉或者音频特征。许多工作都是致力于应用数据挖掘或者基于知识的技术来把视频分类（比如运动、新闻等），然后再进一步划分成子目录（比如划分为足球、篮球等目录）。Zhou 和 Kuo[37]对这些为视频分析搭建智能系统的方法做了总结。



## 18.7 其他格式的查询

利用音频或者音视频相结合的信息来更好地理解多媒体信息的内容是一个非常吸引人的研究方向。Wang 等人[38]对使用音频和视频提示方面进行了介绍。He 等人[39]对教师授课时幻灯片的理解和导航加以介绍, 这些工作基于讲课者花在每张幻灯片上的时间以及讲课者的语调而进行。还有其他一些有意思的方法, 如通过音频检索[40]和“通过哼唱曲子来检索”[41]。

其他一些特征研究人员关注索引的特征, 比如检索动作、概念、感知、面部表情等。显然, 这个领域处在一个发展与成熟的阶段, MPEG-7 标准的制定(参见第12章)推动了这个领域的发展。

## 18.8 基于内容检索的展望

对基于内容检索最新思想的概览[42]指出了一些现在和未来研究的趋势, 基于下面列举的一些方案来对多媒体数据进行索引、搜索、查询和检索:

1) 利用视频特征来对视频进行检索。这些视频特征有: 图像颜色和对象形状、视频分割、视频关键帧、场景分析、对象结构、运动向量、光流(从计算机视觉的角度而言)、多谱线数据, 以及概括数据的所谓的“签名”(signature)。

2) 时空查询, 比如运行轨迹。

3) 语义特征, 句法描述符。

4) 相关反馈(信息检索的常用技术)。

5) 声音, 特别是在一些文档中使用的说话者的声音信息。

6) 多媒体数据库技术, 如图像的关系数据库。

7) 文本、视觉、语音信息的融合。

8) 自动即时的视频操纵, 多媒体数据库允许用户编辑的功能。

9) 多媒体数据安全、隐藏、认证技术(如水印)。

这个领域的内容非常的丰富, 而且和 MPEG-7 的大致方向相互配合。

在另一个研究方向, 研究人员力图建立一个搜索框架来包含更多的实例, 如所有的“动物”。然后, 在检索关系数据库的时候, 这些搜索轮廓就和数据库查询交流。对利用视觉特征的搜索而言, 智能搜索引擎通过主动学习来理解用户的查询意图[43]。这个方向的研究称做“基于学习的查询”。

另一种方法的目标是理解人们根据一些基本的感知如何定义图像之间的相似性[44]。这个方法中所用的功能是一种所谓的“感知相似度量”, 在一系列指定的相似图像中通过学习找到最佳的特征集合(颜色、纹理等)来获取“相似性”。

## 18.9 进一步探索

关于 CBIR 这个主题有一些不错的书[45, 46, 47, 48]值得参考。

在本书的网站上, 第18章的 Further Exploration 中提供了一些有用的基于内容检索的网址。它们有:

- 在 18.2 节中讨论过的 C-BIRD 的 Java applet 版本。
- 作为艺术品服务器的 QBIC 的一个演示版本。
- 亚历山大数字图书馆、伯克利数字图书馆项目、Photobook、Visual RetrievalWare、VIPER 和 VisualSEEK 的演示版本。
- MediaSite(现在已更名为 Sonic Foundary Media Systems)的演示版本。Informedia 项目为这个商业系统提供了功能强大的搜索引擎。

- NETRA 系统的演示版本。其思想是选择一幅图像，然后在图像上选择片段，再在选择的模型上搜索。
- 描述 Virage 系统相关技术的视频。Virage 为 Alta Vista's Image Search 提供搜索引擎。
- 图 18-19 和图 18-20 中介绍过的关键帧产生技术。
- 一些到数字图像和视频的标准集的链接。这些标准集常用来对检索和视频分割的程序作测试。

## 18.10 练习

1. 特征定位的本质是什么？相比传统的图像分割技术而言，它有哪些优缺点。
2. 证明用于更新的式 18.9 的正确性，即第  $k+1$  次迭代时，父场景  $j$  的离心率  $E_j^{(k+1)}$  可以从第  $k$  次迭代得到的父场景  $j$  和子场景  $i$  的离心率、质心、质量推导出来。（提示： $C_{x,j}^{(k)}$ 、 $C_{y,j}^{(k)}$  分别是质心  $C_j^{(k)}$  的  $x$ 、 $y$  分量。）
3. 试用 VIPER 搜索引擎，在几次迭代中使用相关反馈来精化搜索结果。演示版本中提到了 Gabor 直方图和 Gabor 块。在相关网站上查阅文档来确定这两个名词的含义，并简短地描述它们的使用方法。
4. 试用在 18.9 节中提到的实验性质的图像搜索引擎。有些引擎能够给人很深刻的印象，而大部分的搜索引擎在数据范围较大时会变得不可靠。
5. 设计一个图像描述的文本注解分类法，可以使用 Yahoo! 的分类集开始工作。
6. 调查几个网站的图像标题。你认为文字数据对于确定图像的内容有多大用处？（一般而言，搜索系统使用“取词根”的方法，以便去除词中夹杂的时态、大小写变化和数字，取词根可以取出句子的主干。）
7. 至少列举 3 种音频分析辅助视频检索系统相关任务的方式。
8. 假定我们粗略地定义一个颜色直方图，每个存储单元使用 8 位，其中红色和绿色分别用 3 位存储，剩下 2 位存储蓝色。为这个直方图建立一个合适的结构，并利用你读入的图像填充这个直方图。在本书的网站上提供了读取图像的 Visual C++ 的模板代码（“Sample Code”链接下的 sampleCcode.zip 文件）。
9. 试着建立在 18.2.5 节中介绍的纹理直方图。你可以按照该节中介绍的步骤对一些小图像进行处理。为了方便起见，你可以使用 MATLAB。
10. 描述你如何从一个图像数据库中检索出具有 2D “砌砖模式”（brick pattern）的图像。假定“砖块”的颜色是黄色，砖间“空隙”的颜色是蓝色（必须讨论你的方法的局限性和可能的扩展）。
  - (a) 仅使用颜色。
  - (b) 仅使用基于边缘的纹理度量。
  - (c) 使用颜色、纹理、形状。
11. 静态图像和视频之间最主要的区别就是视频中有运动。视频中 CBR 一个重要的部分就是运动估计（例如运动的速度和方向）。如果视频是 MPEG 视频而不是解压缩的视频，请描述如何从这样一个视频剪辑中估计某个对象（比如说视频中出现的某辆车）的运动。
12. 牛顿指出，颜色是三维的。总体来看，我们使用了很多不同的颜色空间，每个颜色空间都有一个亮度轴和两个基本颜色轴，只是不同颜色空间轴的类型不一样而已。如果使用一个色度的二维空间（如式 (4.7) 所定义的），我们将只能使用这个空间的前两维

$\{x, y\}$ 。为一些图像设计 2 维颜色直方图, 并对它们的颜色直方图相交。将这种方法得到的相似度度量值与使用 3 维的颜色直方图得到的结果相比较, 对不同的分辨率都作一个比较。通过比较结果回答: 一般来说, 我们是否有必要保存所有的三维信息?

13. 使用一些低层次的图像特征来实现一个图像搜索引擎, 例如可以用颜色直方图、颜色矩阵、纹理等特征。建立一个至少包含 500 幅图像(可分为 10 个不同的类别)的图像数据库。分别使用单特征和组合特征进行检索。对每个类别而言, 根据准确率和查全率确定哪种特征组合具有最好的检索性能。

## 18.11 参考文献

- [1] M.M. Fleck, D.A. Forsyth, and C. Bregler, "Finding Naked People," in *European Congress on Computer Vision*, 1996, (2):593-602.
- [2] C.C. Chang and S.Y. Lee, "Retrieval of Similar Pictures on Pictorial Databases," *Pattern Recognition*, 24:675-680, 1991.
- [3] S. Paek, C.L. Sable, V. Hatzivassiloglou, A. Jaimes, B.H. Schiffman, S.F. Chang, and K.R. McKeown, "Integration of Visual and Text Based Approaches for the Content Labeling and Classification of Photographs," in *ACM SIGIR'99 Workshop on Multimedia Indexing and Retrieval*, 1999, 423-444.
- [4] K. Barnard and D.A. Forsyth, "Learning the Semantics of Words and Pictures," in *Proceedings of the International Conference on Computer Vision*, 2001, 2:408-415.
- [5] M.J. Swain and D.H. Ballard, "Color Indexing," *International Journal of Computer Vision*, 7(1):11-32, 1991.
- [6] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-Based Image Retrieval at the End of the Early Years" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1349-1380, 2000.
- [7] Z.N. Li, O.R. Zaïane, and Z. Tauber, "Illumination Invariance and Object Model in Content-Based Image and Video Retrieval," *Journal of Visual Communication and Image Representation*, 10(3):219-244, 1999.
- [8] H. Tamura, S. Mori, and T. Yamawaki, "Texture Features Corresponding to Visual Perception," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-8(6):460-473, 1978.
- [9] A.R. Rao and G.L. Lohse, "Towards a Texture Naming System: Identifying Relevant Dimensions of Texture," in *IEEE Conference on Visualization*, 1993, 220-227.
- [10] R. Jain, R. Kasturi, and B.G. Schunck, *Machine Vision*. New York: McGraw-Hill, 1995.
- [11] M.S. Drew, J. Wei, and Z.N. Li, "Illumination-Invariant Image Retrieval and Video Segmentation," *Pattern Recognition*, 32:1369-1388, 1999.
- [12] M.S. Drew, Z.N. Li, and Z. Tauber, "Illumination Color Covariant Locale-Based Visual Object Retrieval," *Pattern Recognition*, 35(8):1687-1704, 2002.
- [13] B.V. Funt and G.D. Finlayson, "Color Constant Color Indexing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:522-529, 1995.
- [14] D.H. Ballard and C.M. Brown, *Computer Vision*, Upper Saddle River, NJ: Prentice Hall, 1982.
- [15] T.H. Hong and A. Rosenfeld, "Compact Region Extraction Using Weighted Pixel Linking in a Pyramid," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:222-229, 1984.
- [16] D. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition*, 13(2):111-122, 1981.
- [17] P. Gvozdzjak and Z.N. Li, "From Nomad to Explorer: Active Object Recognition on Mobile Robots," *Pattern Recognition*, 31(6):773-790, 1998.

- [18] J. Au, Z.N. Li, and M.S. Drew, "Object Segmentation and Tracking Using Video Locales," in *Proceedings of the International Conference on Pattern Recognition (ICPR 2002)*, 2002, 2:544-547.
- [19] M. Flickner, et al, "Query by Image and Video Content: The QBIC System," *IEEE Computer*, 28(9):23-32, 1995.
- [20] J. Hafner, H.S. Sawhney, W. Equitz, M. Flickner, and W. Niblack, "Efficient Color Histogram Indexing for Quadratic Form Distance Functions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:729-736, 1995.
- [21] W. Niblack, Xiaoming Zhu, J.L. Hafner, T. Breuel, D. Ponceleon, D. Petkovic, M.D. Flickner, E. Upfal, S.I. Nin, S. Sull, B. Dom, Boon-Lock Yeo, A. Srinivasan, D. Zivkovic, and M. Penner, "Updates to the QBIC System," in *Storage and Retrieval for Image and Video Databases*, 1998, 150-161.
- [22] Y. Deng, D. Mukherjee, and B.S. Manjunath, "NETRA-V: Toward an Object-Based Video Representation," in *Storage and Retrieval for Image and Video Databases (SPIE)*, 1998, 202-215.
- [23] C. Carson, S. Belongie, H. Greenspan, and J. Malik, "Blobworld: Image Segmentation Using Expectation-Maximization and its Application to Image Querying," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026-1038, 2002.
- [24] A. Pentland, R. Picard, and S. Sclaroff, "Photobook: Content-Based Manipulation of Image Databases," in *Storage and Retrieval for Image and Video Databases (SPIE)*, 1994, 34-47.
- [25] F. Liu and R.W. Picard, "Periodicity, Directionality, and Randomness: Wold Features for Image Modeling and Retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:722-733, 1996.
- [26] R.W. Picard, T. P. Minka, and M. Szummer, "Modeling User Subjectivity in Image Libraries," in *IEEE International Conference on Image Processing*, 1996, 2:777-780.
- [27] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra, "Relevance Feedback: A Power Tool for Interactive Content-Based Image Retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):644-655, 1998.
- [28] A. Hampapur, A. Gupta, B. Horowitz, and C.F. Shu, "The Virage Image Search Engine: An Open Framework for Image Management," in *Storage and Retrieval for Image and Video Databases (SPIE)*, 1997, 188-198.
- [29] Y. Ishikawa, R. Subramanya, and C. Faloutsos, "Mindreader: Querying Databases through Multiple Examples," in *24th International Conference on Very Large Data Bases, VLDB*, 1998, 433-438.
- [30] H.Y. Bang and T. Chen, "Feature Space Warping: An Approach to Relevance Feedback," in *International Conference on Image Processing*, 2002, 1:968-971.
- [31] Y. Lu, C. Hu, X. Zhu, H. Zhang, and Q. Yang, "A Unified Framework for Semantics and Feature Based Relevance Feedback in Image Retrieval Systems," in *Eighth ACM International Conference on Multimedia*, 2000, 31-37.
- [32] R. Brunelli, O. Mich, and C.M. Modena, "A Survey on the Automatic Indexing of Video Data," *Journal of Visual Communication and Image Representation*, 10:78-112, 1999.
- [33] S.F. Chang, et al., "VideoQ: An Automated Content Based Video Search System Using Visual Cues," in *Proceedings of ACM Multimedia 97*, 1997, 313-324.
- [34] D. Bordwell and K. Thompson, *Film Art: An Introduction*, New York: McGraw-Hill, 1993.
- [35] F. Dufaux, "Key Frame Selection to Represent a Video," in *International Conference on Image Processing*, 2000, 2:275-278.
- [36] M.S. Drew and J. Au, "Video Keyframe Production by Efficient Clustering of Compressed Chromaticity Signatures," In *ACM Multimedia 2000*, 2000, 365-368.

- W. Zhou and C.C.J. Kuo, *Intelligent Systems for Video Understanding*, Upper Saddle River, NJ: Prentice-Hall PTR, 2002.
- [38] Y. Wang, Z. Liu, and J.C. Huang, "Multimedia Content Analysis Using Both Audio and Visual Clues," *IEEE Signal Processing Magazine*, 17:12–36, 2000.
- [39] L. He, E. Sanocki, A. Gupta, and J. Grudin, "Auto-Summarization of Audio-Video Presentations," in *ACM Multimedia*, 1999, 1:489–498.
- [40] E. Wold, T. Blum, D. Keislar, and J. Wheaton, "Content-Based Classification, Search, and Retrieval of Audio," *IEEE Multimedia*, 3:27–36, 1996.
- [41] N. Kosugi, Y. Nishihara, T. Sakata, M. Yamamuro, and K. Kushima, "A Practical Query-by-Humming System for a Large Music Database," in *ACM Multimedia*, 2000, 333–342.
- [42] S. Basu, A. Del Bimbo, A.H. Tewfik, and H. Zhang, "Special Issue on Multimedia Database," *IEEE Transactions on Multimedia*, 4(2):141–143, 2002.
- [43] I.J. Cox, M.L. Miller, T.P. Minka, T.V. Papathomas, and P.N. Yianilos, "The Bayesian Image Retrieval System, Pichunter: Theory, Implementation and Psychological Experiments," *IEEE Transactions on Image Processing*, 9(1):20–37, 2000.
- [44] B. Li, E. Chang, and C.T. Wu, "DPF — A Perceptual Distance Function for Image Retrieval," in *IEEE International Conference on Image Processing*, 2002, 2:597–600.
- [45] G. Lu, *Multimedia Database Management Systems*, Norwood, MA: Artech House Publishing, 1999.
- [46] A. Del Bimbo, *Visual Information Retrieval*, San Francisco: Morgan Kaufmann, 1999.
- [47] M. Lew, ed., *Principles of Visual Information Retrieval*, Berlin: Springer-Verlag, 2001.
- [48] V. Castelli and L.D. Bergman, eds., *Image Databases: Search and Retrieval of Digital Imagery*, New York: Wiley, 2002.



# 索引

索引中的页码为英文原书页码, 与书中页边标注的页码一致。

- A-law (A 律), 134, 137, 148
  - compander (压缩扩展器), 205
- $\mu$ -law ( $\mu$ 律), 134, 137, 148, 374
  - compander (压缩扩展器), 205
- 2D mesh (网格), 349
  - geometry coding (几何编码), 349
  - motion coding (运动编码), 352
  - object coding (对象编码), 349
- 2D object animation (2D 对象动画), 352
- 3D model-based coding (基于模型的 3D 编码), 354
- 3D polygon mesh (3D 多边形网格), 356
- 3G, 488, 489, 491, 496, 506
  - G3G (Global 3G) (全球第三代移动通信系统), 489
- AC (Alternate Current) (交流), 209
- Access network (接入网), 439, 464
  - FTTC (Fiber To The Curb) (光纤到社区), 440
  - FTTH (Fiber To The Home) (光纤到家庭), 440
  - HFC (Hybrid Fiber-Coax) cable network (混合光纤/同轴) 电缆网络, 432, 440
  - satellite distribution (卫星传送), 440
  - terrestrial distribution (地面传送), 440
- Access point (接入点), 479
- Active pixel (活动像素), 116
- Active video line (活动视频线), 116
- AD (Analog-to-Digital) converter (模数) 转换器, 135, 158
- Adaptive compression algorithms (自适应压缩算法), 176
- Adaptive Huffman coding (自适应赫夫曼编码), 176
- Adobe Photoshop, 38
  - alpha channel (Alpha 通道), 39
  - magic wand tool (魔术棒工具), 39
- Adobe Premiere, 37
  - timeline window (时间轴窗口), 37
  - transition (切换), 40
- ADPCM (Adaptive Differential Pulse Code Modulation, 自适应差分脉冲编码调制), 147, 158, 374, 376
- ADSL (Asymmetric Digital Subscriber Line, 非对称数字用户线路), 429
- Affine transform (仿射转换), 353
- Alias (假), 128-130
- AM (Amplitude Modulation, 调幅), 426
- AMPS (Advanced Mobile Phone System, 高级移动电话系统), 480
- Analog video (模拟视频), 113
- Animation (动画), 16
  - 3D Studio Max, 16
  - Java 3D, 16
  - Maya, 16
  - RenderMan, 16
  - Softimage, 16
- Anti-aliasing (抗锯齿), 128
  - filter (滤波器), 129
- APC (Adaptive Predictive Coding, 自适应预测编码), 158
- Arithmetic coding (算术编码), 187, 345
- Aspect ratio (画面比例), 116, 121, 123
- ATM (Asynchronous Transfer Mode, 异步传送模式), 428, 435, 459
  - AAL (ATM Adaptation Layer) AAL (ATM 适配层), 437
  - cell (信元), 437
  - layer (层), 437
  - NNI (Network-Network Interface, 网络-网络接口), 437
  - UNI (User-Network Interface, 用户-网络接口), 437
- ATM Adaptation Layer (AAL) ATM 适配层 (AAL), 460
- ATV (Advanced TV, 高级电视), 319
- Audio 音频
  - filtering (滤波), 136
  - formats (格式), 136
- Audio and visual objects (AV 对象), 332
- Audio compression standard (音频压缩标准)
  - G.711, 376
  - G.721, 374, 376

- G. 723, 374, 376
- G. 723.1, 386, 388
- G. 726, 374, 376
- G. 727, 374
- G. 728, 389
- G. 729, 386, 389
- AVI (Audio Video Interleave, 音频视频交叉), 37
- BAB (Binary Alpha Block, 二元 $\alpha$ 块), 344
- Band-limited signal (限带信号), 128
- Band-limiting filter (带限滤波器), 136
- Band-pass filter (带通滤波器), 136, 142, 149
- Bandwidth (带宽), 136
- BAP (Body Animation Parameters, 身体动画参数), 356
- Baseline JPEG JPEG (基本系统), 261
- BDP (Body Definition Parameters, 身体定义参数), 356
- Bi-level image compression standards, (二值图像压缩标准) 参见 JBIG, 282
- BIFS (Binary Format for Scenes, 场景的二进制格式), 333
- Bilinear interpolation (双性插值), 317, 326
- Binary Shape Coding (二元形状编码), 344
- Binary tree (二叉树), 171
- Bitmap (位图), 61
- Bitplane (位平面), 61
- Block-based coding (基于块的编码), 335
- Bluetooth (蓝牙), 493
- BMP (BitMap), 78
- Body object (身体对象), 356
- Buffer management (缓冲区管理)
  - optimal plan (优化设计), 473
  - peak bit-rate (峰值码率), 472, 473
  - prefetch buffer (预取缓冲区), 472
- C-BIRD, 513
  - GUI, 514
  - search by illumination invariance (按光源恒常性查找), 519
  - search by object model (按对象模型查找), 520
- Cable modem (电缆调制解调器), 432, 440
- CAE (Context-based Arithmetic Encoding, 基于上下文的算术编码), 344
- Camera system (照相机系统), 86
- CBIR (Content-Based Image Retrieval, 基于内容的图像检索), 513
  - current CBIR system (当前的 CBIR 系统), 533
- CCIR (Consultative Committee for International Radio, 国际无线电咨询委员会), 120
- CCITT (International Telegraph and Telephone Consultative Committee, 国际电报电话咨询委员会), 150
- CDMA (Code Division Multiple Access, 码分多址), 481, 486
  - cdma2000, 489, 490
  - cdmaOne, 490
- CELP (Code Excited Linear Prediction) (码激励线性预测), 383
  - adaptive codebook (自适应码本), 384
  - LSF (Line Spectrum Frequency, 线谱频率), 387
  - LSP (Line Spectrum Pair, 线谱对), 386
  - LTP (Long Time Prediction, 长时预测), 384
  - stochastic codebook 随机码本, 387
  - STP (Short Time Prediction, 短时预测), 384
- Chroma subsampling (色度二次采样), 120
- Chromaticity (色度), 92
  - diagram (图), 91
- Chrominance (色度), 104
- CIF (Common Intermediate Format, 公共中间格式), 122
- Clustering (聚类), 68
- CMY 减色系统 (Cyan 青, Magenta 品红, Yellow 黄), 101
- CMYK, 102
- Codec (多媒体数字信号编解码器), 168
- Coder mapping (编码器映射), 148
- Codeword (码字), 167
- Color (颜色)
  - cycling (循环), 68
  - density (密度), 516
  - histogram (直方图), 66, 514
  - layout (布局), 516
  - lookup table devising (查找表设计), 68
  - monitor specification (监视器规范), 94
  - picker (选择器), 68
  - primaries (原色), 89
  - subcarrier (副载波), 117
  - texture (纹理), 517
- Color science (颜色科学), 82
  - light and spectra (光和光谱), 82
  - spectra sensitivity of the eye (眼睛的光谱灵敏度), 84
  - visible light (可见光), 82
- Color-matching function (颜色匹配函数), 89



- Commission Internationale de L'Eclairage (CIE, 国际照明协会), 89
- Component video (分量视频), 112
- Composite video (复合视频), 112, 113
- Compression (压缩)
- lossless (无损), 64, 76, 167
  - lossy (有损), 64, 76, 167, 253
  - ratio (率), 76, 168
  - speech (语音), 148
- Compressor function (压缩函数), 205
- Cones (视锥细胞), 84, 85
- Context modeling (上下文建模), 277
- Continuous Fourier Transform (CFT, 连续傅里叶变换), 218
- Continuous Wavelet Transform (CWT, 连续小波变换), 223, 227
- CRT (Cathode Ray Tube, 阴极射线管), 87
- CSMA/CD (Carrier Sense Multiple Access with Collision Detection, 载波侦听多路访问/冲突检测), 432, 439
- CSS (Cascading Style Sheets, 级联样式表), 11
- DA (Digital-to-Analog) converter, 数/模转换器, 135
- DAI (DMIF Application Interface, DMIF 应用程序接口), 463
- Datagram (数据报), 423
- dB (Decibel, 分贝), 131
- DC (Direct Current, 直流), 209
- De-interlacing (去除交错), 114
- Decoder mapping (解码器映射), 148
- Dictionary-based coding (基于字典的编码), 181
- Differential coding (差分编码), 150, 191
- DiffServ (Differentiated Service, 区分服务), 446
- Digital audio (数字音频), 15
- coding of (编码), 147
  - Cool Edit, 15
  - Pro Tools, 15
  - quantization and transmission (量化和传输), 147
  - Sound Forge, 15
- Digital library (数字图书馆), 167
- Digitization of sound (声音数字化), 126
- DCT, Discrete Cosine Transform, 离散余弦变换, 207, 209, 214
- 1D, 208
  - 2D, 207, 208, 253
  - basis function (基函数), 209, 216
- Discrete Fourier Transform (DFT, 离散傅里叶变换), 218
- Discrete Wavelet Transform (DWT) (离散小波变换 (DWT)), 223, 230
- Dispersion (散射), 82
- Distortion measure (失真量度), 199
- MSE (Mean Square Error, 均方差), 199
  - PSNR (Peak Signal to Noise Ratio, 峰值信噪比), 200
  - SNR (signal-to-noise ratio, 信噪比), 200
- Dithering (抖动), 62, 68
- dither matrix (抖动矩阵), 63
  - ordered dither (有序抖动), 63
- DM (Delta Modulation, 增量调制), 157
- adaptive (自适应), 158
  - uniform (均匀), 157
- DMIF (Delivery Multimedia Integration Framework, 多媒体传送集成框架), 462
- DMT (Discrete Multi-Tone, 离散多音频), 430
- Dominant color enhancement (主颜色增强), 525
- DPCM (Differential Pulse Code Modulation, 差分脉冲编码调制), 147, 151, 154, 260
- Dreamweaver, 51
- DV video (Digital Video, 数字视频), 119, 120
- DVB (Digital Video Broadcasting, 数字视频广播), 440, 464
- DVB-MHP (Multimedia Home Platform, 多媒体家庭平台), 464
- DVD, 319, 320, 412, 415
- Dynamic range (动态范围), 151
- EBCOT (Embedded Block Coding with Optimized Truncation)
- EBCOT (优化截断嵌入式块编码), 241, 267, 269, 273, 275
- EDTV (Enhanced Definition TV, 扩展清晰度电视, 参见 HDTV), 124
- Entropy (熵), 168
- coding (编码), 169, 171
- Error concealment (错误隐藏), 424, 501
- Error Resilient Entropy Coding (EREC, 错误修复熵编码), 499
- Ethernet (以太网), 432
- 10-Gigabit Ethernet (10G 以太网), 439
  - Fast Ethernet (快速以太网), 433
  - Gigabit Ethernet (千兆以太网), 438
- Euler's formula (欧拉公式), 218
- Excited purity (激发纯度), 93

- EXIF (Exchange Image File, 交换图像文件), 77
- Expander function (扩展函数), 205
- Extended Huffman coding (扩展的赫夫曼编码), 176
- Extended padding (扩展填充), 340
- EZW (Embedded Zerotree Wavelet) (EZW (嵌入式零树小波)), 241, 242, 244
- FAP (Face Animation Parameter, 脸部动画参数), 355
- Fax Standards (传真标准)
- G3, 282
  - G4, 282
- FDDI (Fiber Distributed Data Interface, 光纤分布式数据接口), 433
- FDMA (Frequency Division Multiple Access, 频分多址), 480
- FDP (Face Definition Parameter, 脸部定义参数), 355
- Feature localization (特征定位), 523
- FM (Frequency Modulation, 调频), 137, 139, 426
- Forward Error Correction (FEC, 前向纠错), 503, 504
- Fourier transform (傅里叶变换), 222
- Frame buffer (帧缓存), 61, 100
- Frame-based coding (基于帧的编码), 335
- FTP (File Transfer Protocol, 文件传输协议), 449, 463
- Gamma correction (伽马校正), 87, 88, 97, 100
- Gamut (色域), 95
- printer (打印机), 102
- Gaussian distribution (高斯分布), 249
- Generalized Markup Language (GML, 通用标记语言), 9
- GIF (Graphics Interchange Format, 图形交换格式), 71
- animation (动画), 16
  - color map (色图), 73
  - GIF87, 73
  - GIF89, 73
  - screen descriptor (屏幕描述符), 71
- Global motion compensation (全局运动补偿), 348
- GPRS (General Packet Radio Service, 通用分组无线业务), 482
- GPS (Global Positioning System, 全球定位系统), 485, 496
- Granular distortion (细粒失真), 202
- Graphics (图形), 15
- Fireworks editing (Fireworks 编辑), 15
  - Freehand editing (Freehand 编辑), 15
  - Illustrator editing (Illustrator 编辑), 15
- Graphics animation file (图形动画文件), 77
- FLC, 77
  - GL, 77
  - IFF, 77
  - Quicktime, 77
- Gray-level (灰度)
- image (图像), 192
  - intensity (亮度), 169
- Grayscale (灰度色标), 75
- Grayscale shape coding (灰度形状编码), 346
- GOV (Group of Video Object Plane, 视频对象平面组 GOV), 335
- GSM (Global System for Mobile communications, 全球移动通信系统), 481, 482, 491
- GSTN (General Switched Telephone Network, 全球交换电话网络), 456
- H. 261, 288
- bitstream (位流), 301
  - block layer (块层), 303
  - encoder and decoder (编码器和解码器), 298
  - formats supported (支持格式), 296
  - GOB (Group of Blocks) layer (GOB (块组)层), 301
  - inter-frame (p-frame) coding (帧内 (p 帧) 编码), 297
  - intra-frame (I-frame) coding (帧间 (I 帧) 编码), 297
  - macroblock layer (宏块层), 302
  - picture layer (图像层), 301
  - quantization (量化), 297
  - step size (步长), 297
- H.263, 288, 303
- motion compensation (运动补偿), 304
  - optional coding mode (可选编码模式), 305
  - PB-frame (PB 帧), 306
- H.263+, 307
- H. 263++, 307
- H. 264, 357
- Baseline Profile (基本规范), 360
  - deblocking filter (去块效应滤波), 360
  - Extended Profile (扩展规范), 361
  - I-prediction (I 预测), 359
  - Main Profile (主要规范), 360
  - P-prediction (P 预测), 359

- H. 26L, 357
- H. 323, 456
- Half-pixel precision (半像素精度), 304
- Halftone printing (网板打印), 63
- Harmonics (和声), 128
- Hartley (哈特利), 168
- HDTV (High Definition TV, 高清晰度电视), 122, 124
- Hierarchical JPEG (分级 JPEG), 263
- Homogeneous coordinate system (齐次坐标系) 353
- Horizontal retrace (水平回扫), 114
- HTML (Hyper Text Markup Language, 超文本标记语言), 10, 35, 56, 71
- HTTP (HyperText Transfer Protocol, 超文本传输协议), 9, 422, 457
- Huffman coding (赫夫曼编码), 173
  - optimality (最优性), 175
  - prefix property (前缀性质), 175
  - procedure (过程), 177
  - tree (树), 177
- Human vision (人的视觉), 84
- Hybrid excitation vocoder (混合激励声音合成器), 389
  - MBE (Multi-Band Excitation, 多频带激励), 389
  - MELP (Multiband Excitation Linear Predictive, 多频带激励线性预测), 386, 391
- Hypermedia (超媒体), 7
- Hypertext (超文本), 7
- IDCT (inverse Discrete Cosine Transform, 逆离散余弦变换), 208
- IETF (Internet Engineering Task Force, 因特网工程任务组), 425, 448, 463
- IGMP (Internet Group Management Protocol, 因特网组管理协议), 449
- Image (图像)
  - 24-bit (24 位), 64
  - 8-bit (8 位), 65
  - data type (数据类型), 64
  - descriptor (描述符), 513
  - Fireworks editing (Fireworks 编辑), 15
  - formation (图像的形式), 85
  - histogram (直方图), 169
  - monochrome (单色), 61
  - Photoshop editing (Photoshop 编辑), 15
  - resolution (分辨率), 61
  - retrieval (检索), 511
- Information theory (信息论), 168
- Intellectual Property Management and Protection (IPMP, 知识产权管理和保护), 369
- Interactive TV (ITV, 交互式电视), 464
- Interlaced scanning (隔行扫描), 113
- Interlacing (隔行), 71
- Internet telephony (因特网电话), 455
- IP (Internet Protocol, 网际协议), 424, 447
- IP-multicast (IP 多播), 447
- IPv4 (IP version 4, IP 版本 4), 424, 448
- IPv6 (IP version 6, IP 版本 6), 425, 447
- ISDN (Integrated Services Digital Network, 综合业务数字网络), 427
- JBIG (Joint Bi-level Image Experts Group, 联合二值图像专家组), 282
- JBIG2, 282
- Jitter (抖动), 444, 462
- JVT (Joint Video Team, 联合视频小组), 参见 H.264, 357
- JPEG (Joint Photographic Experts Group, 联合图像专家组), 75, 253, 255
  - DCT, 255
  - entropy coding (熵编码), 259, 261
  - main steps (主要步骤), 253
  - mode (模式), 262
  - quantization (量化), 255
  - zigzag scan (Z 字扫描), 259
- JPEG-LS, 277
- JPEG2000, 265
- KLT (Karhunen-Loève transform) (Karhunen-Loève 变换), 207
- LAB color model (LAB 颜色模型), 98
- LAN (Local Area Network) (LAN (局域网)), 424, 430
- Latency (延迟), 443, 461
- Locales (场景)
  - definition (定义), 523
  - locale generation (场景生成), 526
  - object matching (对象匹配), 529
  - object modeling (对象建模), 528
  - texture analysis (纹理分析), 528

- tile classification (块分类), 524
- video locales (视频场景), 533
- LOCO-I (LOW COMPLEXITY LOSSLESS COMPRESSION for Images, 图像的低复杂度无损压缩), 277
- Lookup Table (LUT, 查找表), 66-68
- Lossless image compression (无损图像压缩), 191
- Lossless JPEG (无损 JPEG), 193, 264
  - encoder (编码器), 194
  - predictor (预测器), 193
- Lossy image compression (有损图像压缩), 199
- Low-pass filter (低通滤波器), 136
- LPC (Linear Predictive Coding, 线性预测编码), 380, 388
  - LPC-10, 381
- Luminance (亮度), 104
- LZW (Lempel-Ziv-Welch), 71, 73, 78, 181
- Macroblock (宏块), 289
- Macromedia Director, 40
  - 3D Sprite (3D 分镜), 45
  - animation (动画), 42
  - control (控制), 43
  - Lingo script (Lingo 脚本), 44
  - object (对象), 46
- Macromedia Flash, 46
  - animation (动画), 48
  - symbol (符号), 47
  - window (窗口), 46
- MAN (Metropolitan Area Network, 城域网), 430
- MBone, 448
- Mean absolute difference (平均绝对误差), 290
- Media objects (媒体对象), 332
- Media-on-Demand (MOD, 媒体点播), 464, 465
- Median-cut algorithm (中值区分算法), 69
- MIDI (Musical Instrument Digital Interface, 乐器数字化接口), 139
  - banks (音色库), 140
  - channel (通道), 140
  - channel messages (通道消息), 143
  - channel mode (通道模式), 145
  - channel pressure (channel pressure 消息), 144
  - conversion to WAV (转换到 WAV), 147
  - key pressure (key pressure 消息), 144
  - keyboard (键盘), 139
  - patch (编配程序), 140
  - system messages (系统消息), 143, 146
  - velocity (速度), 140, 144
  - voice messages (声部消息), 144
- MiniDV, 117
- MMR (Modified Modified Read) algorithm (MMR (修订修改读) 算法), 344
- MMS (Multimedia Messaging Service, 多媒体消息服务), 507
- Model-based coding (基于模型的编码), 283
- Modem (调制解调器), 426
- Motion Compensation (MC, 运动补偿), 288, 289, 337
  - backward prediction (后向预测), 290
  - forward prediction (前向预测), 290
- Motion estimation (运动估计), 288
- Motion JPEG, 262
- Motion vector (运动向量), 290
  - 2D logarithmic search (2D 对数搜索), 291
  - hierarchical search (分层搜索), 293
  - sequential search (顺序搜索), 290
- MPEG (Moving Pictures Experts Group, 运动图像专家组), 312
- MPEG audio compression (MPEG 音频压缩), 395
  - bark (音频单位), 402
  - bit allocation (位的分配), 409
  - bit reservoir (位池), 412
  - critical band (临界频带), 400
  - equal-loudness curves (等响度曲线), 396
  - frequency masking (频率遮掩), 396, 398
  - MDCT (Modified Discrete Cosine Transform, 修订离散余弦变换), 411
  - MNR (Mask-to-Noise Ratio, 遮掩噪声比), 410
  - MP3 (MP3), 405, 411, 412
  - MPEG layers (MPEG 的层), 405
    - Layer 1 (第 1 层), 405
    - Layer 2 (第 2 层), 405, 410
    - Layer 3 (第 3 层), 405, 411
  - psychoacoustics (心理声学), 395
  - scale factor band (比例因子频带), 412
  - SMR (Signal-to-Mask Ratio, 信号遮掩比), 409
  - temporal masking (时间遮掩), 403
  - threshold of hearing (听觉阈值), 398
- MPEG working model (MPEG 工作模型)

- SM (Simulation Model, 仿真模型), 363
- TM (Test Model, 测试模型), 363
- VM (Verification Model, 验证模型), 363
- XM (eXperimentation Model, 实验模型), 363
- MPEG-1 (MPEG-1), 312
  - B-frame (B 帧), 313
  - bitstream (位流), 318
  - block layer (块层), 319
  - D-frame (D 帧), 319
  - differences from H.261 (与 H.261 的区别), 315
  - Group of Pictures (GOPs) layer (GOP (图片组) 层), 318
  - I-frame (I 帧), 313
  - macroblock layer (宏块层), 319
  - motion compensation (运动补偿), 313
  - P-frame (P 帧), 313
  - performance of (性能), 317
  - picture layer (图片层), 318
  - prediction (预测), 313
  - quantization (量化), 316
  - sequence layer (序列层), 318
  - slice (宏块片), 315
  - slice layer (宏块片层), 319
- MPEG-2 (MPEG-2), 319
  - alternate scan (交替扫描), 322
  - data partitioning (数据划分), 329
  - differences from MPEG-1 (与 MPEG-1 的区别), 329
  - hybrid scalability (混合可伸缩性), 328
  - interlaced video (隔行扫描视频), 320
  - modes of prediction (预测模式), 321
  - profile (规格), 320
  - Program Stream (程序流), 329
  - SNR scalability (SNR 可伸缩性), 324
  - spatial scalability (空间可伸缩性), 326
  - temporal scalability (时间可伸缩性), 326
  - Transport Stream (传送流), 329
- MPEG-2AAC (Advanced Audio Coding) (MPEG-2AAC (高级音频编码)), 412
  - low complexity profile (低复杂度规格), 413
  - main profile (主要规格), 413
  - PQF (Polyphase Quadrature Filter) bank (PQF (多相积分滤波器) 组), 414
  - scalable sampling rate profile (可伸缩采样率规格), 413
- TNS (Temporal Noise Shaping) tool (TNS (时域噪声整形)), 413
- MPEG-21 (MPEG-21), 369, 415
- MPEG-4 (MPEG-4), 332, 335
  - binary shape (二元形状), 343
  - body object (身体对象), 354
  - Delaunay mesh (Delaunay 网络), 350
  - face object (人脸对象), 354
  - gray-scale shape (灰度形状), 344
  - Part 10, also see H.264 (Part 10, 参见 H.264), 358
  - uniform mesh (均匀网络), 349
  - wavelet coding (小波编码), 346
- MPEG-4 AAC (Advanced Audio Coding) (MPEG-4 AAC (高级音频编码)), 414
  - BSAC (Bit-Sliced Arithmetic Coding, 比特分片算术编码), 414
  - perceptual coders (感知编码器), 414
  - perceptual noise substitution (感知噪声置换), 414
  - SAOL (Structured Audio Orchestra Language, 结构化音频命令语言), 415
  - SNHC (Synthetic/Natural Hybrid Coding) (SNHC (合成/自然混合编码)), 414
  - structured coder (结构编码器), 414
  - TTS (Text-To-Speech, 文语转换), 414
- MPEG-7 (MPEG-7), 361
  - Description Definition Language (DDL, 描述定义语言), 368
  - Description Scheme (DS, 描述模式), 365
  - descriptor (描述子), 363
- MPEG-7 audio (MPEG-7) 音频, 415
- MPEG-J (MPEG-J), 333
- MPEGlet (MPEGlet), 333
- Multimedia (多媒体), 3
  - history of (历史), 5
  - projects (项目), 5
  - research (研究), 4
  - tools (工具), 14
- Multimedia authoring (多媒体编著), 17, 20
  - Authorware (Authorwarer), 17
  - automatic authoring (自动编著), 33
  - Director (Director), 17
  - Dreamweaver (Dreamweaver), 51

- Flash (Flash), 17
- Quest (Quest), 17
- technical issues (技术问题), 31
- tools (工具), 37
- Multimedia metaphors (多媒体模式), 21
  - card/scripting metaphor (卡片/脚本模式), 22
  - cast/score/scripting metaphor (角色/乐谱/脚本模式), 22
  - frames metaphor (框架模式), 22
  - hierarchical metaphor (层次模式), 21
  - iconic/flow-control metaphor (图标/流控制模式), 21
  - scripting metaphor (脚本模式), 21
  - slide show metaphor (幻灯片显示模式), 21
  - Multimedia presentation (多媒体展现), 25
    - graphics style (图形风格), 25
    - sprite animation (分镜(子图像)动画), 27
    - video transition (视频切换), 28
- Multimedia production (多媒体作品), 23
  - flowchart phase (流程图阶段), 24
  - prototyping and testing (构建原型和测试), 24
  - storyboard (故事项), 24
- Multipath Fading (多径衰减), 494
  - Rayleigh fading (Rayleigh 衰减), 494
  - Rician fading (Rician 衰减), 494
- Multiple Protocol Label Switching(MPLS) (多重协议标记交换(MPLS)), 446
- Multiplexing (多路复用), 425
  - DWDM (Dense WDM, 稠密波分多路复用), 426
  - FDM (Frequency Division Multiplexing, 频分多路复用), 425
  - TDM (Time Division Multiplexing, 时分多路复用), 426
  - WDM (Wavelength Division Multiplexing, 波分多路复用), 426
  - WWDM (Wideband WDM, 宽带波分多路复用), 426
- Multiresolution analysis (多分辨率分析), 223, 225
- Multisampling (多采样), 142
- Munsell color naming system (Munsell 颜色命令系统), 100
- MUSE (Multiple sub-Nyquist Sampling Encoding, 多子奈奎斯特采样编码), 122
- Music sequencing (音序), 14
  - Cakewalk, 14
  - Cubase, 14
  - Soundedit, 15
- Network (网络)
  - Application Layer (应用层), 422
  - connection-oriented (面向连接的), 423
  - connectionless (无连接), 424
  - Data Link Layer (数据链路层), 421
  - Network Layer (网络层), 421
  - Physical Layer (物理层), 421
  - Presentation Layer (表示层), 422
  - Session Layer (会话层), 421
  - Transport Layer (传输层), 421
  - Network Address Translation (NAT, 网络地址翻译), 425
  - NMT (Nordic Mobile Telephony, 北欧移动电话系统), 480
  - NTSC (National Television System Committee, 美国国家电视系统委员会), 88,94,312
  - Nyquist (奈奎斯特)
    - frequency (频率), 129
    - rate (比率), 128,149,158
    - theorem (定理), 128,427
  - Object-based visual coding (基于对象的视觉编码), 335
  - OC (Optical Carrier, 光载波器), 429
  - Octave (八度音程), 128
  - OFDM (Orthogonal Frequency Division Multiplexing, 正交频分复用), 492
  - ONU (Optical Network Unit, 光纤网络单元), 440
  - Optimal Work-Ahead Smoothing Plan (最佳超前流畅设计), 473
  - Orthogonal (正交), 216
  - Orthonormal (标准正交), 216
  - basis (基), 230
  - OSI (Open Systems Interconnection, 开放式系统互连), 421
  - Out-of-gamut color (超光谱颜色), 95
  - Packet fragmentation (包分割), 424
  - Padding (填充), 338
    - Horizontal repetitive padding (水平重复填充), 339
    - Vertical repetitive padding (垂直重复填充), 339
  - PAINT (PAINT), 78
  - PAL (Phase Alternating Line, 逐行倒相), 94,119,312
  - Palette animation (调色板动画), 68
  - PAN (Personal Area Network, 个人区域网络), 432
  - PCM (Pulse Code Modulation, 脉冲编码调制),

- 147-150,374
- PCN (Personal Communications Network, 个人通信网络), 479
- PCS (Personal Communication Services, 个人通信服务), 479
- PDA (Personal Digital Assistant, 个人数字助理), 479,507
- PDC (Personal Digital Cellular, 个人数字蜂窝), 479
- PDF (Portable Document Format, 便携式文档格式), 78
- Pel (Pel), 61
- Perceptual nonuniformity (感知不一致), 134,161,400,446
- Pervasive computing (普适计算), 492, 507
- Phase Modulation (相位调制), 426
- PICT, 78
- Pitch (音高), 128, 139
- Pixel (像素), 61
- Pixel clock (点时钟), 116
- PNG (Portable Network Graphics, 便携式网络图形), 76  
alpha-channel ( $\alpha$ 通道), 77
- Polyphony (多音), 140
- Post Compression Rate Distortion (PCRD, 压缩后比例失真), 272
- PostScript, 78
- POTS (Plain Old Telephone Service, 普通老式电话服务), 455, 482
- PPM, 79
- Precision (准确率), 541
- Predictive coding (预测编码), 154  
lossless (无损), 151  
lossy (有损), 151
- Prioritized delivery (具有优先级的发送), 447
- Profile (规格), 356
- Progressive JPEG (渐进式 JPEG), 262
- Progressive scanning (渐进扫描), 113
- PSTN (Public Switched Telephone Network, 公共交换电话网络), 434, 455, 456
- QAM (Quadrature Amplitude Modulation, 正交幅度调制), 426, 429, 440
- QCIF (Quarter-CIF, 四分之一帧像素 CIF), 122
- QPSK (Quadrature Phase-Shift Keying, 正交相移键控), 426, 440
- Quadrature modulation (正交调制), 117
- Quality of Service (QoS) (服务质量 (QoS)), 424, 443, 445, 451, 453, 461-463
- QoS for IP (IP 服务质量), 446
- Quantifying search results (量化搜索结果), 541
- Quantization (量化), 128, 255  
decision boundary (判定边界), 148  
distortion (失真), 155  
error (误差), 132, 155  
linear format (线性格式), 133  
noise (噪声), 132  
nonuniform (非均匀), 133  
nonuniform quantizer (非均匀量化器), 204  
companded quantizer (压缩扩展量化器), 205  
Lloyd-Max quantizer (Lloyd-Max 量化器), 204  
reconstruction level (重构层), 148  
uniform (均匀), 133  
uniform scalar quantizer (均匀标量量化器) 201  
midrise (中高), 201  
midtread (中宽), 201  
vector quantization (向量量化), 206  
codebook (码本), 206
- Quantizer (量化器)  
backward adaptive (后向自适应), 376  
Jayant, 377  
Lloyd-Max, 154
- Querying on video (视频查询), 542
- Rate-distortion theory (比率失真理论), 200  
rate-distortion function (比率失真函数), 200
- Recall (查全率), 541
- Reference frame (参考帧), 290
- Region of Interest (ROI, 感兴趣区域), 266, 275
- Relevance feedback (相关反馈), 539
- Retina (视网膜), 84
- RGB, 100, 363
- RGB to CMY (RGB 到 CMY), 101
- RLC (Run-length Coding, 游长编码), 171, 259
- Rods (柱状细胞), 84
- RSVP (Resource ReSerVation Protocol, 资源预留协议), 451
- RTCP (Real Time Control Protocol, 实时控制协议), 451
- RTP (Real-time Transport Protocol, 实时传输协议), 449
- RTSP (Real Time Streaming Protocol, 实时流协议), 453, 454
- Run-length Encoding (RLE, 游长编码), 78, 259

- RVLC (Reversible Variable Length Code) (RVLC (可逆变长编码)), 498
- S-Video, 112, 113
- SA-DCT (Shape Adaptive DCT) (形状自适应 DCT), 341, 342
- Sampling (采样), 127
  - alias frequency (假频), 129
  - fixed rate (固定采样率), 129
  - folding frequency (折线频率), 129
  - frequency (频率), 127
  - nonuniform (非均匀), 128
  - rate (率), 128
  - true frequency (真实频率), 129
  - uniform (均匀), 128
- SAP (Session Announcement Protocol, 会话通告协议), 458
- SDH (Synchronous Digital Hierarchy, 同步数字体系), 429
- SDP (Session Description Protocol, 会话描述协议), 459
- SDTV (Standard Definition TV, 标准清晰度电视), 参见 HDTV, 124
- SEAM (Scalable and Efficient ATM Multicast, 可扩展的有效 ATM 多播), 462
- SECAM (Système Electronique Couleur Avec Memoire, 顺序传送与存储彩色电视系统), 94, 119
- Sequencer (音序器), 139, 143
- Sequential JPEG (顺序 JPEG), 262
- Set-top Box (STB) (机顶盒 (STB)), 464, 472
- Shannon-Fano algorithm (香农-凡诺算法), 171
- Shape coding (形状编码), 343
- SIF (Source Input Format, 源输入格式), 312
- SIP (Session Initiation Protocol, 会话启动协议), 456
- SMART (Shared Many-to-many ATM Reservations, 共享多对多 ATM 预留), 462
- SMIL (Synchronized Multimedia Integration Language, 同步多媒体集成语言), 12
- SMPTE (Society of Motion Picture and Television Engineers, 美国电影电视工程师学会), 88, 94
  - SMPTE-170M, 88
- SMS (Short Message Service, 短消息服务), 482
- SNR (Signal-to-Noise Ratio, 信噪比), 131, 430
- SONET (Synchronous Optical Network, 同步光纤网络), 428,
- Sound (声音)
  - card (卡), 139
  - digitization (数字化), 127
  - wave (波), 126
- Spatial (空间)
  - domain (域), 191
  - frequency (频率), 75, 105
  - redundancy (冗余), 254, 288
- Spectral Power Distribution (SPD, 光谱能量分布), 参见 spectrum, 82
- Spectrophotometer (分光光度计), 82
- Spectrum (光谱), 82
  - locus (轨迹), 92
- SPiHT (Set Partitioning in Hierarchical Trees, 层次树的集合划分), 241, 247
- Spread spectrum (扩频), 483
  - direct sequence (直频), 484
  - frequency hopping (跳频), 483
- Sprite (子图像), 347
  - coding (编码), 347
- SQNR (Signal to Quantization Noise Ratio, 信号量化噪声比), 131
- sRGB (Standard RGB, 标准 RGB), 89, 108
- Standard Generalized Markup Language (SGML, 标准通用标记语言), 9
- Static texture coding (静态纹理编码), 346
- STM (Synchronous Transport Module, 同步传送模块), 429
- Streaming media (流媒体), 453
  - streaming audio (音频流), 453
  - streaming video (视频流), 453
- STS (Synchronous Transport Signal, 同步传输信号), 429
- Subband (子带), 234
- Surface spectral reflectance (表面光谱反射), 85
- Switching (交换), 434
  - cell relay (ATM) (信元中继 (ATM)), 435
  - circuit switching (电路交换), 434
  - frame relay (帧中继), 435
  - packet switching (包交换), 434
- Sync skew (同步偏离), 444
- Synthesizer (合成器), 139, 142
- Synthetic object coding (合成对象编码), 349
- Synthetic sound (合成声音), 137
- TACS (Total Access Communication System, 全访问通信系



- 统), 480
- Target frame (目标帧), 289
- TCP (Transmission Control Protocol, 传输控制协议), 423
  - Acknowledgement (ACK) (确认 (ACK)), 423
  - retransmission timeout (重传超时), 423
  - window (窗口), 423
- TCP/IP protocol (TCP/IP 协议), 422
- TDMA (Time Division Multiple Access, 时分多址), 481
- Television systems (电视系统)
  - NTSC (National Television System Committee, 国家电视委员会系统), 113, 116
  - PAL (Phase Alternating Line, 逐行倒相), 119
  - SECAM (Système Electronique Couleur Avec Memoire, 顺序传送与存储彩色电视系统), 119
- Temporal redundancy (时间冗余), 289
- Texture (纹理)
  - analysis (分析), 517
  - coding (编码), 341
  - layout (布局), 517
- TIFF (Tagged Image File Format, 标记图像文件格式), 77
- Timbre (音色), 140
- Token ring (令牌环), 433
- Transducer (传感器), 126
- Transform coding (变换编码), 207
- Transmission rate control (传输率控制), 473
- Tristimulus values (三色值), 91
- Two Sided Geometric Distribution (TSGD) (双面几何分布 (TSGD)), 281
- u-law (also see  $\mu$ -law) ( $\mu$ 律), 134, 135, 137
- Ubiquitous computing (普适计算), 492, 507
- UDP (User Datagram Protocol, 用户数据报协议), 424
- Variable-length Coding (VLC, 变长编码), 167, 171, 306
- Vertical retrace (垂直回扫), 114
- Video bitrates (视频码率), 459
  - ABR (Available Bit Rate, 可用码率), 459
  - CBR (Constant Bit Rate, 恒定码率), 459, 472
  - nrt-VBR (non real-time Variable Bit Rate, 非实时可变码率), 459
  - rt-VBR (real-time Variable Bit Rate, 实时可变码率), 459
  - UBR (Unspecified Bit Rate, 未定码率), 459
  - VBR (Variable Bit Rate, 可变码率), 459, 472
- Video broadcasting (视频广播), 465
  - Greedy Equal Bandwidth Broadcasting (GEBB, 贪婪等带宽广播), 467
  - harmonic broadcasting (谐波广播), 468
  - pagoda broadcasting (宝塔广播), 470
  - pyramid broadcasting (金字塔广播), 466
  - staggered broadcasting (交错广播), 465
  - stream merging (流合并), 470
- Video card (视频卡), 61
- Video conferencing (视频会议), 295, 303, 360
- Video editing (视频编辑), 15
  - After Effects, 16
  - Final Cut Pro, 16
  - Premiere, 15
- Video Object (VO, 视频对象), 334
- Video Object Layer (VOL, 视频对象层), 334
- Video Object Plane (VOP, 视频对象平面), 335
- Video Signals (视频信号), 112
- Video-object Sequence (VS, 视频对象序列), 333
- Video-on-Demand (VOD, 视频点播), 443, 464, 465
- Vocoder (声音合成器), 378
  - CELP (Code Excited Linear Prediction, 码激励线性预测), 383
  - channel vocoder (通道声音合成器), 378
  - formant vocoder (共振峰声音合成器), 380
  - LPC (Linear Predictive Coding, 线性预测编码), 380
  - phase insensitivity (相位不敏感), 378
- VoIP (Voice over IP, IP 电话), 455
- VRML (Virtual Reality Modelling Language, 虚拟现实建模语言), 51
  - animation and interaction (动画和交互), 54
  - history (历史), 51
  - overview (概述), 51
  - shapes (形状), 52
  - specifics (规范), 54
- W3C (world Wide Web Consortium, 万维网协会), 8, 368
- WAN (Wide Area Network, 广域网), 424, 430, 434
- WAP (Wireless Application Protocol, 无线应用协议), 493
- Wave table (波形表), 137
  - data (数据), 142

- file (文件), 139
- synthesis (合成), 138
- Wavelength dominant (主波长), 93
- Wavelet (小波), 222
  - admissibility condition (准入条件), 229
  - analysis filter (分析滤波), 232
  - basis (基), 230
  - biorthogonal (双正交), 232
  - compact support (紧致支持), 232
  - mother wavelet (母小波), 229
  - synthesis filter (合成滤波器), 232
- WCDMA (Wideband CDMA, 宽带 CDMA), 488, 489, 491
- Weber's Law (韦伯定律), 133
- White-point correction (白点矫正), 96
- Wireless LAN (WLAN, 无线局域网), 432, 488, 492
- IEEE 802.11, 492
- IEEE 802.11a, 492
- IEEE 802.11b, 492
- IEEE 802.11g, 493
- Wireless PAN (WPAN, 无线个人区域网络), 432
- WMF (Windows Meta File, WMF 格式), 60, 78
- WWW (World Wide Web, 万维网), 8, 71, 332, 462
- XML (Extensible Markup Language, 可扩展标记语言), 11, 361, 368
- XYZ to RGB (XYZ 到 RGB), 97
- YCbCr, 107, 120, 363
- YIQ, 104, 105, 254
- YUV, 104, 254

# 多媒体技术教程

本书出自资深教师之手，内容取自课堂上讲述的实际素材，适合作为计算机科学和工程专业学生的教材。本书从多媒体编著和数据表现、多媒体数据压缩以及多媒体通信和检索三个层面对多媒体涉及的基本概念、基本原理和基本技术进行了详细介绍。

## 本书特点：

- 介绍多媒体创作工具，例如，音乐序列发生器、图像和视频编辑器，XML和SMIL等流行语言，以及Director、Flash、VRML等程序。
- 图形/图像/视频/音频数据表示，包括颜色模型、HDTV、MIDI和音频编码。
- 数据、图像、视频和音频的压缩格式和标准，包括无损压缩和有损压缩。
- 多媒体网络，考虑了 QoS、VoIP、实时媒体点播和无线网络上的多媒体。
- 数字图书馆中基于内容的检索。

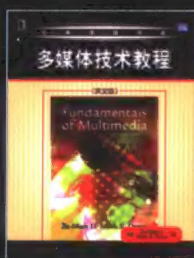
本书有一个相关的网站<http://www.cs.sfu.ca/mmbook>，为教师和学生提供多媒体课程的完整素材和主题以及大量的资源。

## 作者简介

**Ze-Nian Li** 教授现任加拿大温哥华的Simon Fraser大学计算机学院的院长，他还是该校视觉和媒体实验室的主任。

**Mark S. Drew** 现为Simon Fraser大学计算机学院的副教授。

两人均已发表了80多篇关于多媒体及相关领域的论文。



英文影印版  
ISBN 7-111-14686-7  
定价：55.00元



[www.PearsonEd.com](http://www.PearsonEd.com)



ISBN 7-111-19975-8



华章图书

上架指导：计算机/多媒体

华章网站 <http://www.hzbook.com>

网上购书：[www.china-pub.com](http://www.china-pub.com)

投稿热线：(010) 88379604

购书热线：(010) 68995259, 68995264

读者信箱：[hzjsj@hzbook.com](mailto:hzjsj@hzbook.com)

ISBN 7-111-19975-8

定价：42.00 元